RL-TR-94-216, Volume II (of two)
Final Technical Report
December 1994

NEURAL NETWORK FALSE ALARM FILTER

Raytheon Company

F. Aylstock, L. Elerin, J. Hintz, C. Learoyd, and R. Press

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

Rome Laboratory
Air Force Materiel Command
Griffiss Air Force Base, New York

This report has been reviewed by the Rome Laboratory Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RL-TR-94-216, Volume II (of two) has been reviewed and is approved for publication.

APPROVED: Dale W. Richards

DALE W. RICHARDS
Project Engineer

FOR THE COMMANDER: John J. Bart

JOHN J. BART
Chief Scientist, Reliability Sciences
Electromagnetics & Reliability Directorate

If your address has changed or if you wish to be removed from the Rome Laboratory mailing list, or if the addressee is no longer employed by your organization, please notify RL (ERSR) Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document require that it be returned.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1994	Final Sep 92 - Sep 94

4. TITLE AND SUBTITLE
NEURAL NETWORK FALSE ALARM FILTER

5. FUNDING NUMBERS
C - F30602-92-C-0065
PE - 62702F
PR - 2338
TA - 02
WU - 5V

6. AUTHOR(S)
F. Aylstock, L. Elerin, J. Hintz, C. Learoyd, and R. Press

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Raytheon Company
Missile Systems Division
50 Apple Hill Drive
Tewksbury MA 01876-0901

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Rome Laboratory (ERSR)
525 Brooks Road
Griffiss AFB NY 13441-4505

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
RL-TR-94-216, Vol II (of two)

11. SUPPLEMENTARY NOTES
Rome Laboratory Project Engineer: Dale W. Richards/ERSR/(315) 330-3476

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

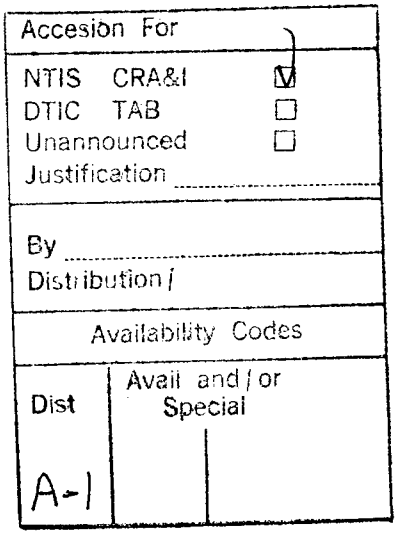
13. ABSTRACT (Maximum 200 words)

This effort identified, developed and demonstrated a set of approaches for applying neural network learning techniques to the development of a real-time built-in test (BIT) capability to filter out false-alarms from the BIT output. Following a state-of-the-art assessment, a decision space of 19 neural network models, 9 fault report causes and 12 common groups of BIT techniques was identified. From this space, 4 unique, high-potential combinations were selected for further investigation. These techniques were subsequently simulated for application to a MILSATCOM system. Detailed analyses of their strengths and weaknesses were performed along with cost/benefit analyses. This study concluded that the best candidates for neural network insertion are new systems where neural network requirements can be included in the initial system design and that a major challenge is the availability or real data for training of the networks. Volume I of this report documents the activities and findings of the effort, including an extensive, annotated bibliography. Volume II contains a tutorial overview of the neural networks, BIT techniques and false alarm causes utilized in the final phases of this study.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Reinforcement Network, Neural Networks, Built-in Test, False Alarm, Backpropagation	96
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

TABLE OF CONTENTS



1. APPENDIX I. NEURAL NETWORK TUTORIALS

This appendix contains the neural network tutorials which were held during the down selection portion of the NNFAF contract. A tutorial was held for each of the network models which were selected (either as primary candidate or alternate) for the NNFAF demonstration approaches. The five neural network models were: Adaptive Resonance Theory 1, Backpropagation, Backpropagation Through Time, Reinforce, and Spatiotemporal Pattern Recognition.

FAF TUTORIAL: ART 1
ADAPTIVE RESONANCE THEORY (1)

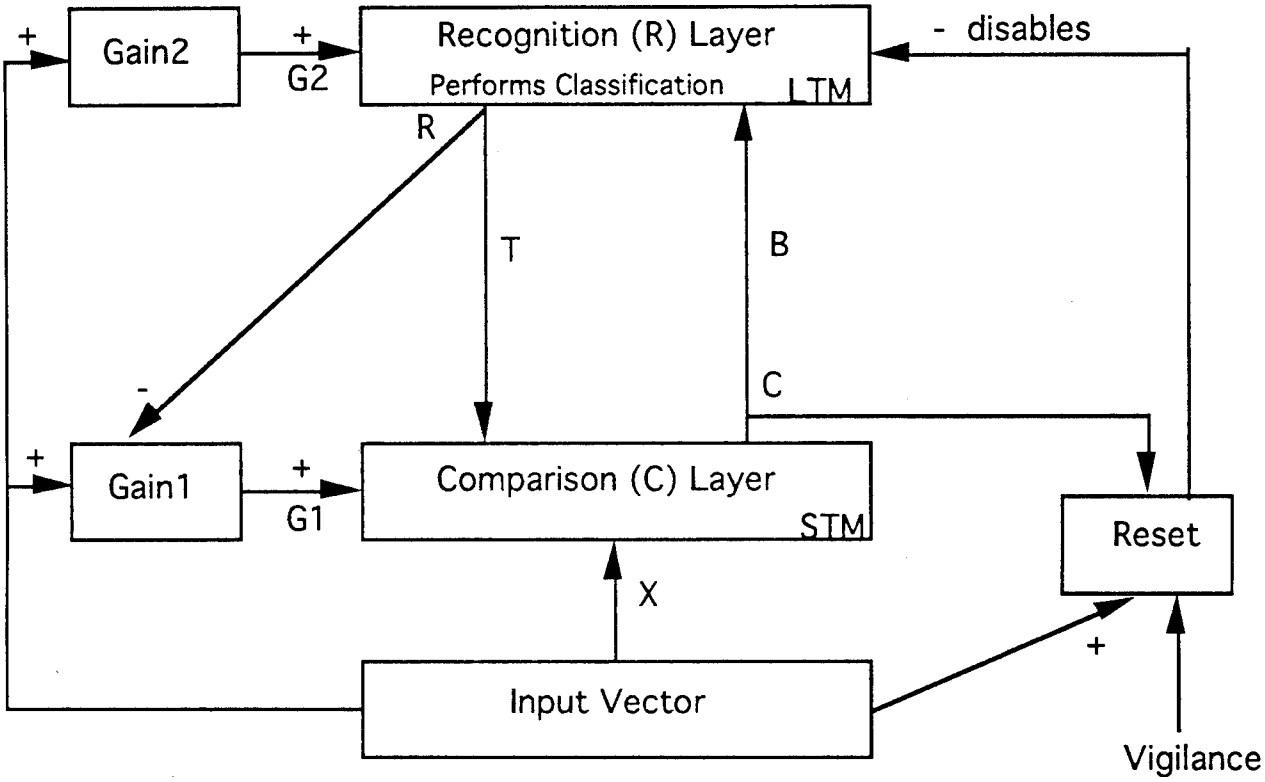
S. Grossberg & G. Carpenter
Boston University

- OVERVIEW
- ARCHITECTURE
- OPERATION
- STRENGTHS/WEAKNESSES
- ISSUES IN NETWORK DESIGN
- EXAMPLE

OVERVIEW

- Unsupervised Learning
- Binary Input
- "SEMI"-Adaptive (on-line) Learning (with NWare TOOL)
- Vector classifier: accepts unknown input vector, classifies according to which stored pattern it most closely resembles
- If input doesn't match any stored pattern, new category is created by storing a pattern like the input vector
- If stored pattern matches input vector within specified tolerance (vigilance), then stored pattern is adjusted (trained) to "add" characteristics of input vector
- NO STORED PATTERN IS EVER MODIFIED IF VIGILANCE IS NOT SATISFIED (not like BP in which any weights can be modified)
- Competitive (winner take all learning)

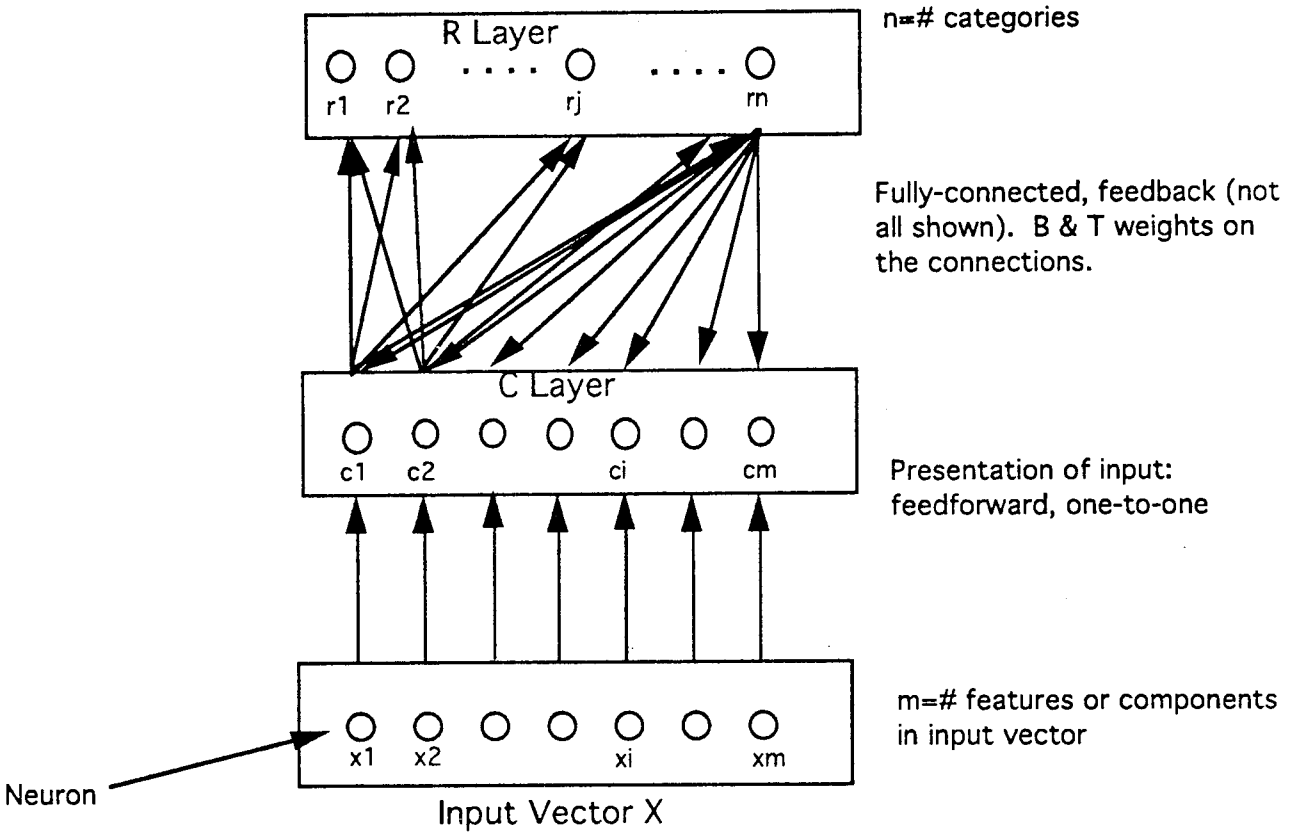
ARCHITECTURE



LTM = Long Term Memory
STM = Short Term Memory
B = Bottom-up weights
T = Top-down weights
R = R layer activation
C = C layer activation

ARCHITECTURE

CONNECTIVITY



ARCHITECTURE

- Input vector X

- Comparison Layer C (short-term memory, stores important features of current pattern)

- Recognition Layer R (long-term memory, stores learned prototype in weight matrix B)

- Full, feedback connections between C and R: bottom-up (B) and top-down (T) weights

- C Layer outputs sent to R layer. No competition.

- R Layer classifies input vector. ONLY ONE R neuron (the one with the weight vector B which best matches the input vector) will fire. The others are inhibited by lateral connections (lateral inhibition).

- Reset: measures closeness between C and X. If they differ by more than vigilance (a real value, 0<vigilance<1), a reset signal is sent to disable the neuron which fired in the R layer.

- Vigilance: controls classification granularity. If high, fine distinction between classes. If low, patterns will be more liberally grouped, less in common but still in the same class.

- Gains: control firing of neurons at each layer and when layers should and should not interact (resonate).

OPERATION

PHASES: Initialization, Recognition, Comparison, Search, Learning

DO FOR every X:
• Initialization: init B, T, vigilance, etc. Go to Recognition.

• Recognition: present input vector X. Compute C activations by 2/3 rule. Send C activations to R layer. Compute R activations. Determine winning R layer neuron. Go to Comparison.

• Comparison: Send feedback from R to C. Set new value of C (C=X LAND R). Compare C to X. If closeness < vigilance, produce R reset, go to Search to look for better match, ELSE go to Learning.

• Search: repeat Recognition/Comparison UNTIL:
- R layer neuron wins competition and vigilance is satisfied. Go to Learning (found best match); OR
- All committed R layer neurons have been disabled by reset. Create/commit new R layer neuron, set to be like X. END DO

• Learning: (fast learning assumed: input vectors are applied for a long enough period of time so that weights reach their final values). Modify B and T to include the common characteristics of X (the networkhas learned something new about the given class).

OPERATION

- After learning, the T weights are set to C (C = X LAND R), so that they only contain the components of the stored prototype which match the input vector.

- The stored prototype eventually represents the logical intersection of all vectors of that class. The essential / common / minimum features are kept.

STRENGTHS/WEAKNESSES

STRENGTHS:
- unsupervised, don't need to know the answers beforehand
- non-linear separability (not sure of limit)
- solves stability-plasticity dilemma: retains old knowledge while acquiring new
- if patterns close to each other, won't have to store many templates (logical intersection)

WEAKNESSES:
- assumes that patterns that share a greater number of input features should fall into the same category
- order of presentation of inputs will change the way the system reacts
- noise/pattern distortion can cause improper classification
- potential for large storage reqts
- fast in analog h/w, slow in serial digital h/w (sequential search of all patterns for best match)
- may create more than "real" number of classes (this is OK)

ISSUES IN NETWORK DESIGN

TO OVERCOME WEAKNESSES, RECOGNIZE IMPORTANCE OF:

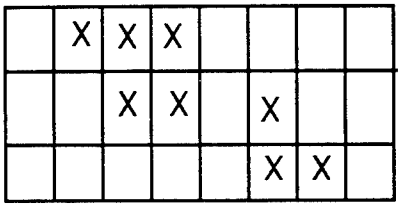
- invariant feature encoding to avoid misclassification due to noise
- feature selection and definition impacts which categories are generated
- number of categories (need to have enough)
- input presentation order - voting scheme not an option
- changing vigilance in real-time to avoid misclassifications

NETWORK DEFINITION:

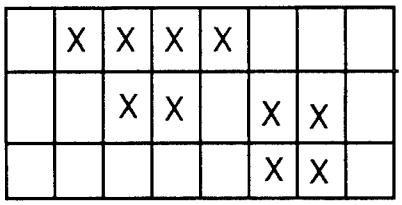
- number of input nodes = number of components in input vector
- number of C layer nodes = number of input nodes
- number of R layer nodes (categories) = some number > the projected number of categories

EXAMPLE

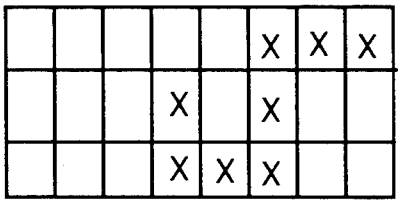
P1



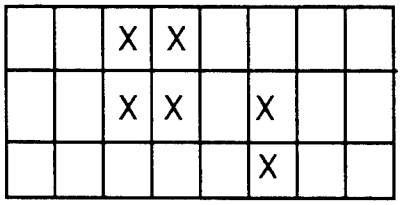
P2



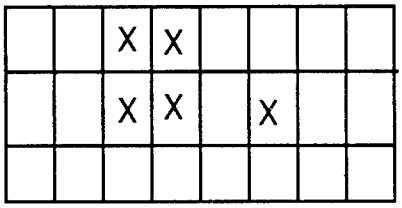
P3



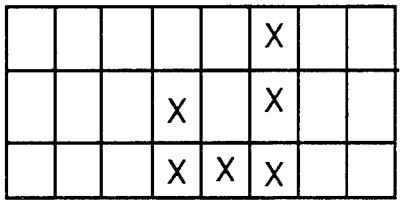
P4



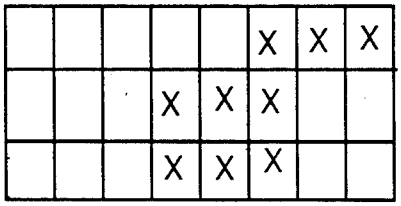
P5



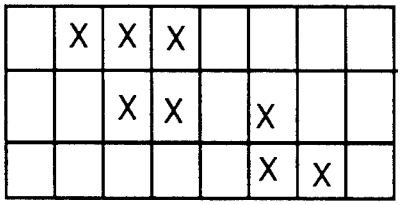
P6



P7



P8



EXAMPLE

CLOSE ENOUGH = differ by LQ 2 pixels

- After P1: Memory contains P1.

- After P2: P2 was close enough to P1 to be in the same class. Since you perform logical intersection of input and stored memory, memory contains P1.

- After P3: P3 is not close enough to P1. Memory contains P1 and P3.

- After P4: P4 is close enough to P1. P1 would be changed to be P4. Memory contains P3 and P4.

- After P5: P5 is close enough to P4. P4 would be changed to be P5. Memory contains P3 and P5.

- After P6: P6 is close enough to P3. P3 would be changed to be P6. Memory contains P5 and P6.

- After P7: P7 is not close enough to any of them. Memory contains P5, P6, P7.

- After P8: P8 is not close enough to any of them. Memory contains P5, P6, P7, P8.

EQUATIONS

Vigilance: $0 < t < 1$

Gain 1: = 1 if any X = 1 and no R = 1, else = 0

Gain 2: = 1 if any X = 1, else = 0

C: C = X if R inactive
C = X land R_j if R_j active

2/3 Rule (C Activation): Each C neuron receives 3 inputs:
- X
- R_j
- Gain 1

Two of these must = 1 in order for C neuron to be active

R Activation: $Net_k = B \cdot C$

R_k = 1 if Net_k > threshold, else = 0

R_j is active only if $R_j \geq \max(R)$

Learning: Only on a match:

$t_{ji} = c_i$

$b_{ij} = \frac{L}{L - 1 + ||C||}$ where L > 1 (usually = 2)

Reset: $\frac{||C||}{||X||} < t$

Init: $t_{ji} = 1$

b_{ij} = random; $0 \leq b_{ij} \leq \frac{1}{m}$ where $m = ||X||$

X = 0, Gain 1 = Gain 2 = 0, R layer output = 0

FAF TUTORIAL: BACKPROPAGATION

Rumelhart, Hinton & WIlliams
(also Parker)

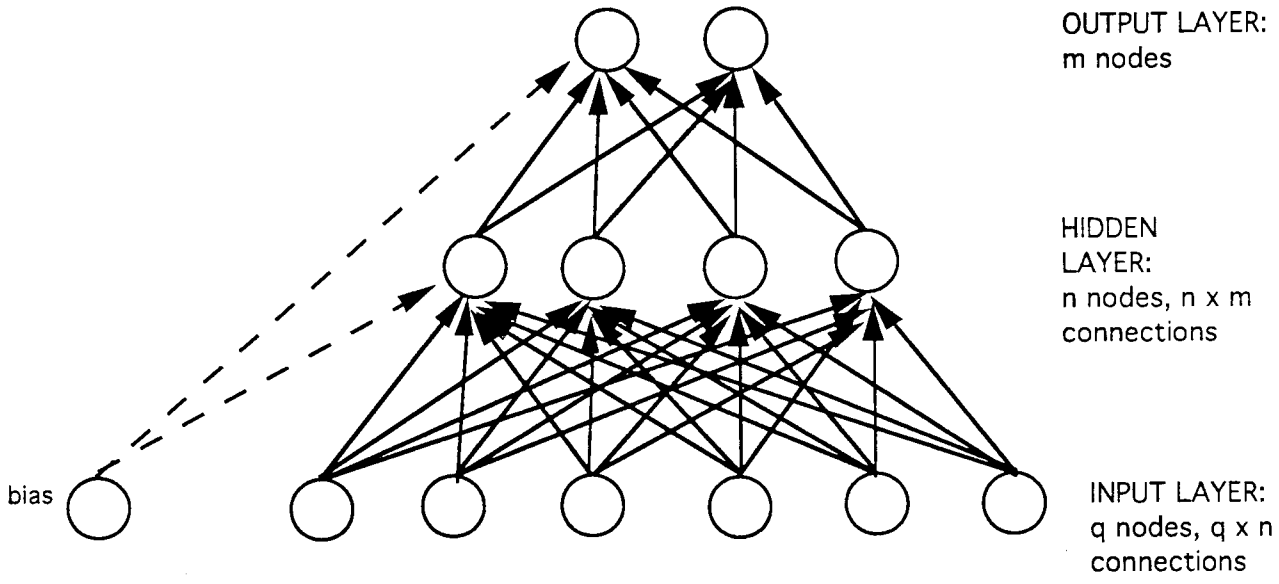
- OVERVIEW
- ARCHITECTURE
- OPERATION
- NETWORK PARAMETERS & TERMS
- STRENGTHS/WEAKNESSES
- ISSUES IN NETWORK DESIGN

OVERVIEW

- Most well-known, widely-used model
- Supervised Learning
- Not limited to binary input
- Multi-layer network, solves non-linearly separable classifications
- Sometimes known as "Generalized Delta Rule"
- Learns an internal representation of the input, as well as learning the output
- Credit Assignment problem: If output is in error, how do you determine which weight (connection) to adjust? Different solution than ART: assumes all nodes are partially responsible for the error. Propagates the output error backward thru the connections, thru all layers, to the input layer, changing ALL weights.
- NOT Competitive (winner take all) learning
- Used for: pattern classification, data compression, noise filtering, signal processing, stock market prediction, converting English text to phonemes, etc.

ARCHITECTURE

FEEDFORWARD CONNECTIVITY



- **Multi-layer: input, hidden, output**
- **At least ONE hidden layer required (usually 1 or 2)**
- **Feedforward, fully connected between adjacent layers; connections have associated weights**

OPERATION

PHASES: Training (learning), Testing (recall)

Training:
• Assign random real-valued weights to each connection

• REPEAT FOR EACH TRAINING DATA SET UNTIL CONVERGENCE OR UNTIL REPETITION LIMIT REACHED:
- run training pattern thru network
- determine error (distance) between the actual value output and the known desired output at each output node
- using a steepest descent algorithm, back propagate this error through the network, adjusting weights. Weights which were further off are updated more.
• At end of training, weights are saved to be used for testing

Testing:
• WEIGHTS ARE NOT CHANGED
• single pass thru each test pattern
• run each pattern thru the network
• the values at the output nodes constitute a classification, with the maximum value corresponding to the best estimate of identification

OPERATION

TRANSFER FUNCTIONS

sigmoid, maps to (0..1) = $\frac{1}{(1+e^{-z})-1}$

tanh, maps to (-1..1) = $\frac{1-e^{-z}}{1+e^{-z}}$

NOTATION:

- a_j = current activation of node j in layer below i (child of i)
- a_i = current activation of node i in layer above j (parent of j)
- w_{ij} = weight on the connection joining node j to node i

THREE PHASES OF TRAINING:

- Present input vector, propagate forward to output layer by calculating activations of nodes upward from input layer to output layer, generate output vector:

$a_i = f(\sum_j w_{ij} a_j)$ where f is transfer function (assume sigmoid)

- **Backpropagate local error (recursive):**

  output units:
  calculate scaled error: $error_i = (t_i - a_i) * a_i (1 - a_i)$
  change weights: $\Delta w_{jk} = L\ (error_j)\ (a_k)$ *k child of j*
  hidden units:
  calculate scaled error:
  $error_j = a_j(1-a_j) * \sum_i error_i\ w_{ij}$
  change weights: $\Delta w_{jk} = L\ (error_j)\ (a_k)$ *k child of j*

- **Update weights:**

  for all units, new $w_{ij} = w_{ij} + \Delta w_{ij}$

NETWORK PARAMETERS/TERMS

• Initialization of Weights: if all weights started at equal value and the solution requires that unequal weights be developed, the system will never learn because all the weight changes will be the same. Init to random values.

• Transfer Function: Why the sigmoid function? Derivative exists (it is continuous); derivative required for gradient descent learning method. Also, the sigmoid derivative can be defined in terms of the sigmoid function itself. (Tanh is the same)

• Learning Rate: In gradient descent, changing the weight assumes that the error surface is locally linear (locally is defined by size of learning rate). It is important to keep learning rate low, to avoid divergent behavior at points of curvature. The ideal situation would be to step by infinitely small increments, but time does not permit this. How to solve this dichotomy?

• Momentum: includes the effect of past weight changes on the current direction of movement in weight space. It is used to avoid large changes in either direction. It allows smaller learning rate but faster learning.

• Epoch: number of iterations per training set (convergence or limit)

STRENGTHS/WEAKNESSES

STRENGTHS:

- small storage reqts
- well-known

WEAKNESSES:

- many variables, trial and error
- slow training, many iterations thru data to convergence, not sure when to stop, not sure it will ever converge (can cycle instead)
- overtraining, can learn "noise"
- local minima

ISSUES IN NETWORK DESIGN

- Which transfer function to use
- Normalization of input
- Number of input nodes = number of components in input vector
- Defining number of hidden nodes (heuristics) - more hidden nodes will increase execution time, but if too small, may miss local minima
- Number of output nodes = number of classes
- Typically each upper layer should have fewer nodes than lower one
- Size must be reasonable (max 200-300 nodes for s/w simulation)
- Momentum (how conservative you are in going down the error slope) - allows smaller learning rate constant with faster learning, but means more storage used to store previous weights
- Storage: including bias, need $(q+1)n + (n+1)m$ weights
- Mix classes when training to avoid shocks
- How to speed up training time: use slightly noisy data, increase size of hidden layer BUT keep size of hidden layer reasonable, use variations of learning algorithms
- What to do if network doesn't learn: start over with new initial weights
- Avoid memorization (keep #hidden nodes > #output nodes)

FAF TUTORIAL: BACKPROPAGATION THROUGH TIME (BPTT)

Rumelhart, Hinton & WIlliams
Werbos
Williams & Zipser

- REVIEW OF BACKPROP
- ARCHITECTURE OF BPTT
- OPERATION OF BPTT
- WEAKNESSES

REVIEW OF BACKPROP

- Most well-known, widely-used model

- Supervised Learning, solves non-linearly separable problems

- Learns an internal representation of the input, as well as learning the output

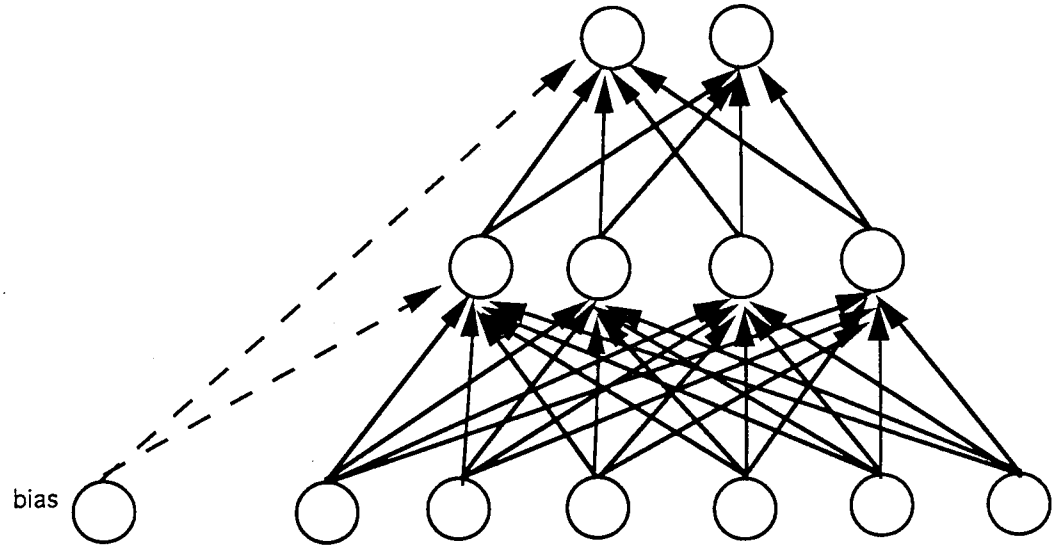
- Credit Assignment problem: If output is in error, how do you determine which weight (connection) to adjust? Assumes all nodes are partially responsible for the error. Propagates the output error backward thru the connections, thru all layers, to the input layer, changing ALL weights.

- Multi-layer: input, hidden, output

- At least ONE hidden layer required (usually 1 or 2)
- Feedforward, fully connected between adjacent layers; connections have associated weights

REVIEW OF BACKPROP

FEEDFORWARD CONNECTIVITY



OUTPUT LAYER:
m nodes

HIDDEN
LAYER:
n nodes, n x m
connections

INPUT LAYER:
q nodes, q x n
connections

REVIEW OF BACKPROP

PHASES: Training, Testing

Training:
• Assign random real-valued weights to each connection

• REPEAT FOR EACH TRAINING DATA SET UNTIL CONVERGENCE OR UNTIL REPETITION LIMIT REACHED:
- run training pattern thru network
- determine error (distance) between the actual value output and the known desired output at each output node
- using a steepest descent algorithm, back propagate this error through the network, adjusting weights. Weights which were further off are updated more.
• At end of training, weights are saved to be used for testing

Testing:
• WEIGHTS ARE NOT CHANGED
• single pass thru each test pattern
• run each pattern thru the network
• the values at the output nodes constitute a classification, with the maximum value corresponding to the best estimate of identification

REVIEW OF BACKPROP

TRANSFER FUNCTIONS

sigmoid, maps to (0..1) = $\frac{1}{(1+e^{-z})-1}$

tanh, maps to (-1..1) = $\frac{1-e^{-z}}{1+e^{-z}}$

THREE PHASES OF TRAINING:

• Present input vector, propagate forward to output layer by calculating activations of nodes upward from input layer to output layer, generate output vector:

$a_i = f(\sum_j w_{ij}\, a_j)$ where f is transfer function (assume sigmoid)

• Backpropagate local error (recursive):

output units:

calculate scaled error: $error_i = (t_i - a_i) * a_i (1 - a_i)$

change weights: $\Delta w_{jk} = L\ (error_j)\ (a_k)$ *k child of j*

hidden units:

calculate scaled error:

$error_j = a_j(1-a_j) * \sum_i error_i\, w_{ij}$

change weights: $\Delta w_{jk} = L\ (error_j)\ (a_k)$ *k child of j*

• Update weights:

for all units, new $w_{ij} = w_{ij} + \Delta w_{ij}$

OVERVIEW OF BPTT

- Temporal supervised learning task: sequence classification

- The input is the sequence to be classified

- The desired output is the correct classification, which is to be produced at the end of the sequence.

- Gradient-based approach: part of the learning algorithm involves computing the gradient of a performance measure, and using the result to determine the weight changes.

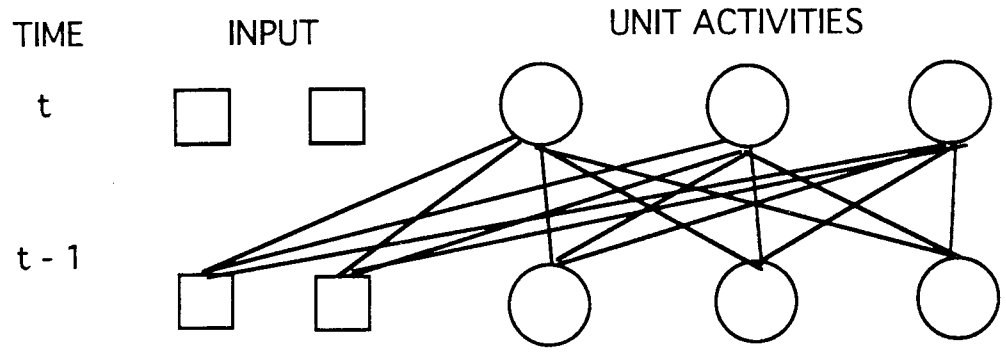
- Performance measure: measure of error between actual & desired output

- Epochwise Operation: network runs from start state to stopping time, then reset to start state for next epoch. Starting states do not have to be the same. Epoch boundary is barrier across which credit assignment should not pass.

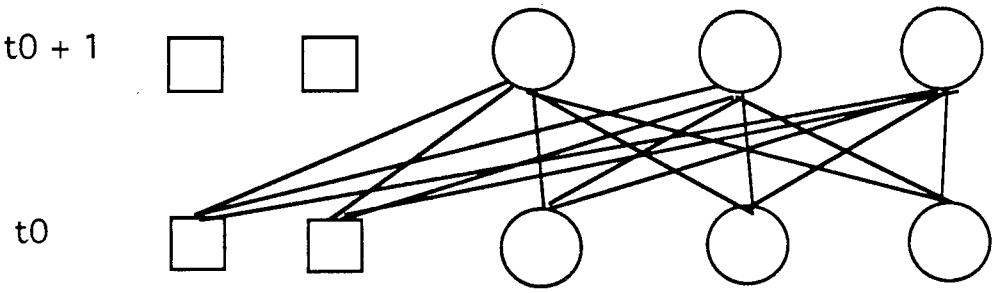
- Epoch Notation: (t0 = start time, t1 = end time)
- Epochwise Learning Algorithm: weight updates are performed only at epoch boundaries, not at every time step

- Assumptions: semilinear units, discrete time

ARCHITECTURE OF BPTT



EACH CONNECTION IS ASSUMED TO HAVE A DELAY OF 1 TIME STEP



OPERATION OF BPTT

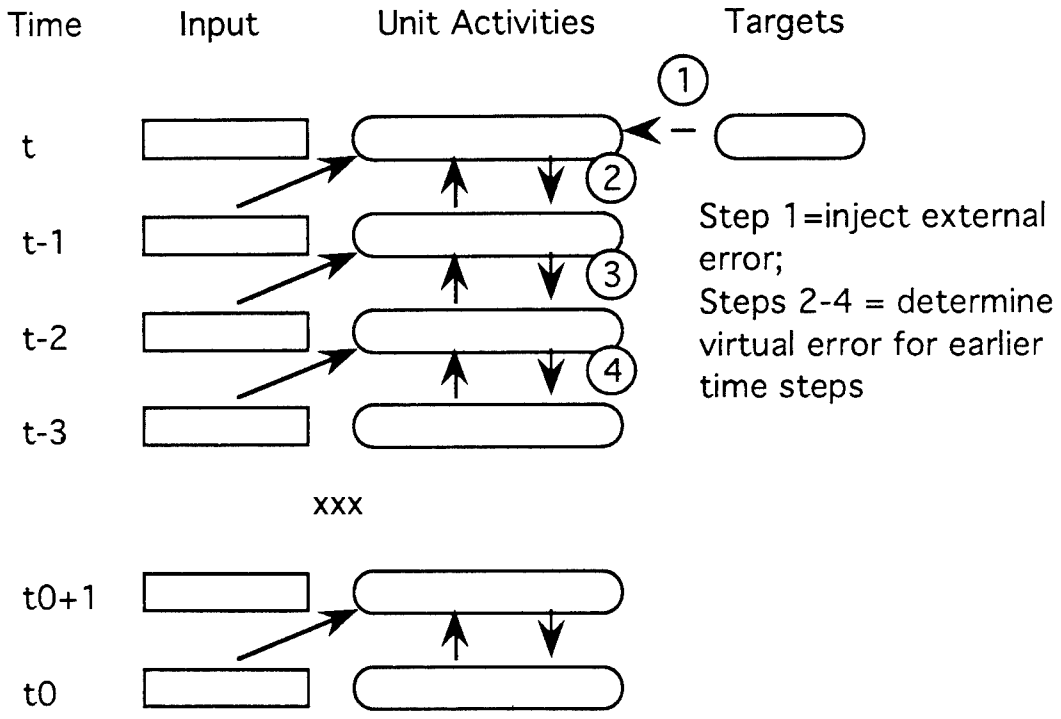
- **Real-Time BPTT:**

Do at each time step t:

1. Add current state of network and current input pattern to a history buffer which stores history of network since time t0

2. Inject error for current time. Backpropagation used to compute all the errors and error derivatives for t0 < ti < t

3. All weights are changed accordingly.

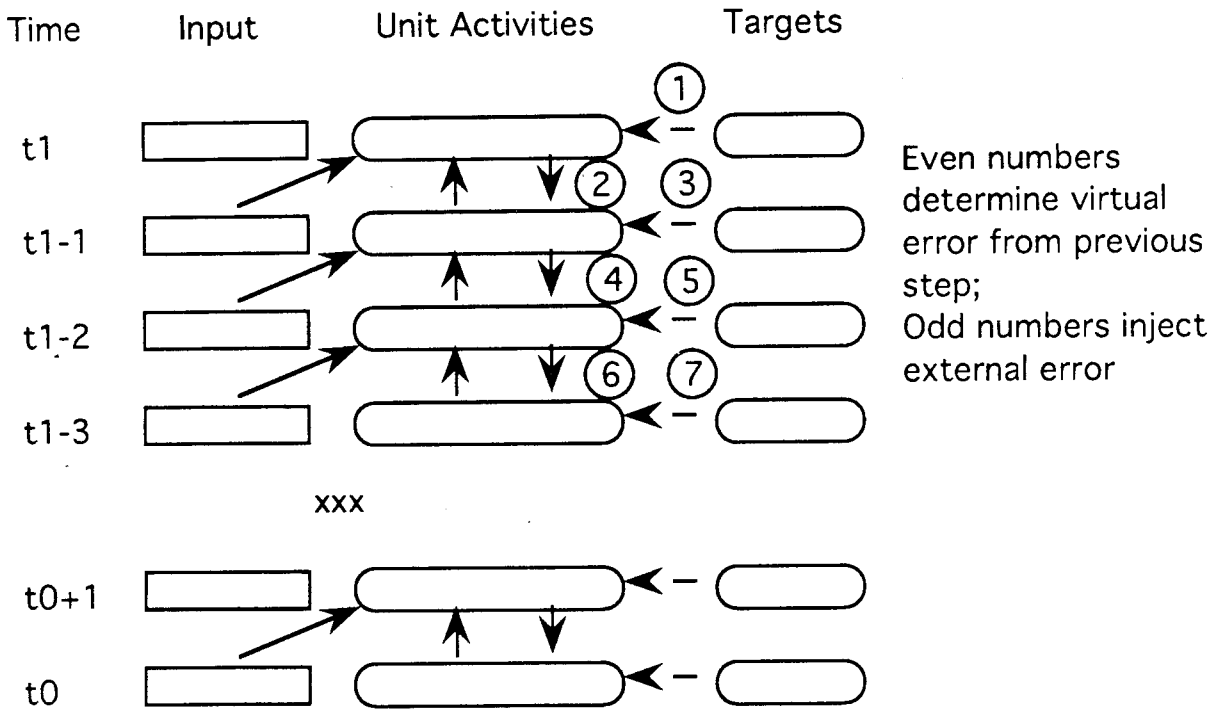


OPERATION OF BPTT

- **Epochwise BPTT**

During each epoch, accumulate the history of network input and network activity, along with history of target output values / history of error. Do at each epoch:

1. **Backpropagation used to compute all the errors and error derivatives for t0 < t < t1**
2. **All weights are changed accordingly.**
3. **Reinitialize network and begin next epoch.**



WEAKNESSES

- **STORAGE REQTS/COMPUTATION TIME: dependent upon selection of time granularity and temporal pattern length**

FAF TUTORIAL: REINFORCE

R. J. WIlliams

- OVERVIEW
- ARCHITECTURE
- REINFORCE ALGORITHMS
- NETWORK ISSUES

OVERVIEW

- DEFINITION OF REINFORCEMENT LEARNING (as distinguished from supervised or unsupervised learning:)

The performance of the entire system is judged on the basis of a single scalar value, called REINFORCEMENT, received from the environment, as its evaluation of system performance.

At one extreme, the signal may have 2 values: success/failure

A more informative signal would have a continuum of values, indicating a graded degree of success

GENERAL OBJECTIVE OF LEARNING: the system must maximize some function of the reinforcement signal

The computation of reinforcement by the environment is problem specific AND IS ASSUMED TO BE UNKNOWN TO THE LEARNING SYSTEM.

OVERVIEW

- ASSOCIATIVE REINFORCEMENT LEARNING:

The environment provides additional information beyond the reinforcement signal itself.
The system learns to ASSOCIATE OUTPUTS WITH INPUTS (INPUT-OUTPUT MAPPING).
The system determines what action to perform (what the OUTPUT should be) based on the additional information from the environment and on the REINFORCEMENT signal.

- Why interesting?

These systems require (for training feedback) a SINGLE SCALAR REINFORCEMENT SIGNAL provided to the entire net.

They statistically move along the gradient of a natural performance measure for these problems (analogous to backprop).

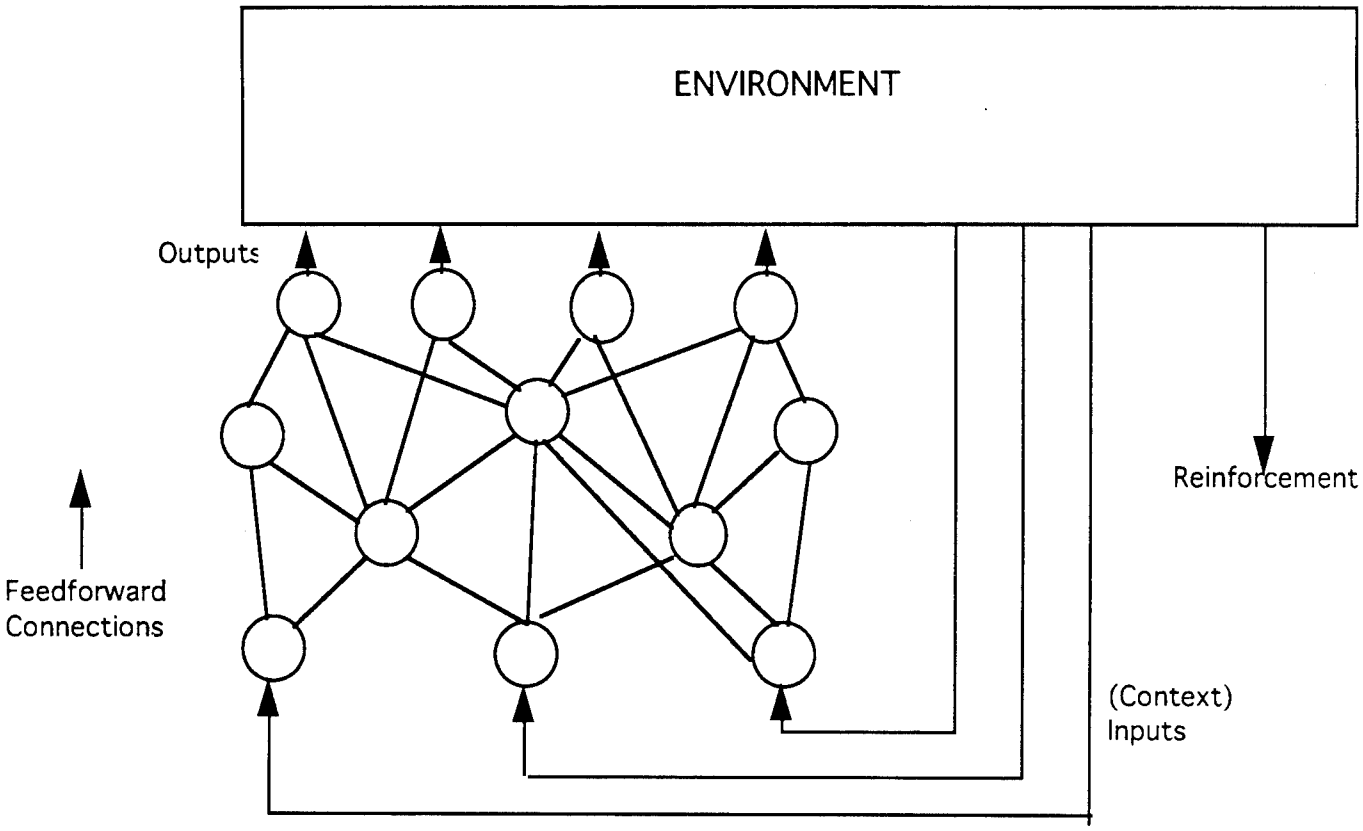
They can be implemented "simply" even in a temporal context.

OVERVIEW

ASSOCIATIVE REINFORCEMENT LEARNING	SUPERVISED LEARNING
Evaluative feedback (system presented with scalar signal)	Instructive feedback (system presented with desired output)
Must discover output: must search all possible actions to discover which is better. Output cannot be a deterministic function of input; the operation of the system has certain random components.	Knows output: no autonomous search capability required

Random operation consistent with theory of stochastic learning automata.

OVERVIEW


ENVIRONMENT
Outputs
Reinforcement
Feedforward
Connections
(Context)
Inputs

OVERVIEW

A network of associative stochastic learning automata and its training environment for a restricted associative reinforcement learning task. In the network setting, individual automata are called UNITS, the vector of actions selected by the network is its OUTPUT, and the context input is called INPUT. The operation of this system consists of the following four phases:

1. The environment picks an input pattern for the network randomly (the distribution of which is assumed to be independent of prior events within the network/environment system).

2. As the input pattern to each unit becomes available, it picks an action randomly according to the distribution of actions corresponding to the particular input pattern. Thus, "activation" passes thru the network from input side to output side.

3. After all the units at the output side have selected their actions, the environment picks an evaluation randomly according to a distribution corresponding to the particular network output pattern chosen and the particular network input.

4. Each unit changes its internal state according to some specific function of its current state, the action just chosen, its input, and the reinforcement. The precise manner in which the reinforcement signal is used by the units depends upon the learning algorithm to be applied. In the simplest case, the reinforcement signal is simply broadcast to all units, but the use of additional units or interconnections designed to help in the learning process is also possible.

- All units receive identical reinforcement.

- Other strategies are possible: adaptively generated, individually tailored reinforcement signals for individual units or groups of units, as a function of current NON-reinforcement environmental input.

- RESTRICTED associative reinforcement learning task: each unit makes exactly one action selection corresponding to each reinforcement value received. The actions (outputs) are independent of prior history, and therefore of time.

ARCHITECTURE

NOTATION for Quasilinear Stochastic Units:

x_i is the input pattern to that unit. The pattern is a tuple whose individual elements are either the outputs of certain other units, or certain inputs from the environment.

y_i is the output of the i^{th} unit in the network. y_i is drawn from a distribution depending upon x_i and the connection weights w_{ij}.

Y_i is the set of possible output values y_i of the i^{th} unit.

X_i is the set of possible values of the input vector x_i to the i^{th} unit.

For each i, $g_i = Pr\{y_i = E|W, x_i\}$, a probability mass function determining the value of y as a function of the weights and the input:

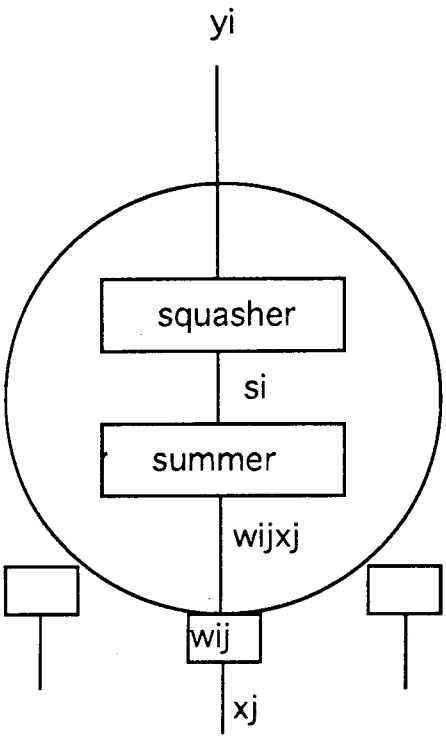
Assume mass function has single parameter p_i,

$p_i = f(s_i)$

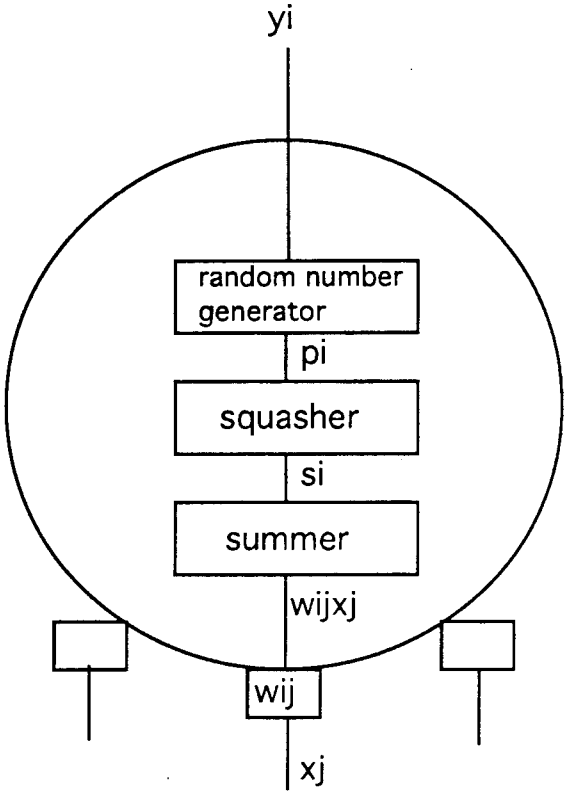
$s_i = \sum w_{ij} x_j$

$f(s_i)$ is usually the sigmoid function

ARCHITECTURE



Deterministic Quasilinear Unit



Stochastic Quasilinear Unit

• Bernoulli Unit: any unit whose purely stochastic component consists of a Bernoulli random number generator, with input to this component representing the Bernoulli parameter p, regardless of the particular nature of the deterministic component of the unit's computation.

REINFORCE ALGORITHMS

- EXPECTED REINFORCEMENT PERFORMANCE CRITERION

The performance measure which will be optimized is the expected value of the reinforcement signal, conditioned on a particular choice of parameters of the learning system (E).

ASSUMPTIONS: stationary distribution of input
inputs are independent from trial to trial
stationary distribution of r

Given these assumptions, E is a well-defined deterministic function WHICH IS UNKNOWN TO THE LEARNING SYSTEM. The learning system must search the parameter space for a point where E is maximum.

ALSO NOTE that since the weight matrix W represents the network parameters, we will be finding the WEIGHTS which maximize E.

REINFORCE ALGORITHMS

- RESTRICTED REINFORCE ALGORITHM: At the end of each trial, r is received by the network and W is adjusted according to the specific learning algorithm.

- Learning algorithm: $\Delta w_{ij} = \alpha_{ij} (r - b_{ij}) e_{ij}$
  where

  α_{ij} is a learning rate factor
  b_{ij} is a reinforcement baseline

  e_{ij} is characteristic eligibility of w_{ij} ($\delta \ln g_i / \delta w_{ij}$)
  $(r-b_{ij})$ is reinforcement offset

Reinforcement baseline is assumed to be conditionally independent of y, given W and x

The Learning Rate is assumed to be non-negative and constant and not dependent upon the input x (but may be dependent upon i and/or j).

REward Increment = Nonnegative Factor x Offset Reinforcement x Characteristic Eligibility

REINFORCE ALGORITHMS

- Just as backprop performs local optimization of an error measure, REINFORCE does essentially the same for the natural performance measure E.

- Associative Reward/Inaction algorithm: Bernoulli quasilinear units with logistic squashing function, constant learning rate and reinforcement baseline = 0:

$$\Delta w_{ij} = \alpha r(y_i - p_i)x_j$$

- REINFORCMENT COMPARISON

This leads to faster and more reliable learning
Rewards actions which lead to better than usual reinforcement and penalizes actions which lead to worse than usual reinforcement.

A prediction of what reinforcement value to expect on a particular trial is used as the basis for comparison.

Prediction is computed as an exponentially weighted average of past reinforcement values. It is adaptive.

For associative tasks, it is desirable to try to predict reinforcement as a function of the input.
$\Delta w_{ij} = \alpha(r - r^{pred})\ (y_i - p_i)\ x_j$, where r^{pred} is the predicted reinforcement for the current input pattern

REINFORCE ALGORITHMS

• EXTENDED REINFORCE ALGORITHMS: Extend algorithm to problems which have temporal credit-assignment component: a network is trained on an episodic basis, where each episode consists of k time steps, during which the units may recompute their outputs and the environment may alter its non-reinforcement input at each time step. A single r value is delivered to the net at the end of each episode.

One way to adapt a network algorithm for temporality is to use the "unfolding in time mapping". The learning algorithm becomes:

$$\Delta w_{ij} = \alpha_{ij} (r - b_{ij}) \sum_{t=1}^{k} e_{ij}(t)$$

where

α_{ij} is a learning rate factor
b_{ij} is a reinforcement baseline independent of y
e_{ij} is characteristic eligibility of w_{ij} ($d\ln g_i/dw_{ij}$) evaluated at time t, depends on the input x to the i^{th} unit at time t-1
$(r-b_{ij})$ is reinforcement offset

The learning rate is assumed to be non-negative and constant.

REINFORCE ALGORITHMS

This algorithm has a "plausible on-line implementation using a single accumulator for each parameter wij in the network." The purpose of this accumulator is to form the eligibility sum, each term of which depends only on the operation of the network as it runs in real time, and not on the reinforcement signal eventually received.

This is in contrast to BPTT, which requires accumulating pairwise products of activations with error signals, requiring large amounts of additional storage which grows linearly with the number of time steps per episode.

REward Increment = Nonnegative Factor x Offset Reinforcement x Cumulative Eligibility

• Informational Connections: may be added to the network.

Signals received on these lines would be used to compute the reinforcement baseline. For example, the reinforcement baseline might try to track the reinforcement received as a function of these informational inputs. A unit may only receive such connections from units on which it has no ultimate influence.

Using this technique might provide more tailored credit assignment; or might help the scaling problems inherent in simpler reinforcement schemes in which all units are reinforced alike.

REINFORCE ALGORITHMS

- Backpropagating Through a Model:

Train a second network, called an internal model, to compute the average reinforcement received as a function of input to and output of the first basic network. The first network must be run in an exploratory mode to cover a sufficiently large portion of the input/output pairs.

After the second network has learned to compute the reinforcement signal provided by the environment, the basic network can be trained by having it hillclimb toward a maximum of the internal reinforcement signal. This can be performed by backpropagation.

The unknown mapping used by the environment to compute the reinforcement is eventually replaced by a known differentiable mapping which provides a reasonable approximation to it.

NETWORK ISSUES

- Where/How is the R signal generated?

- Connectivity: how to connect, how many units to choose, how to layer, is layering meaningful?

- Training vs. testing: don't present reinforcement during testing?

- How to determine learning rate, reinforcement baseline...

- Paper provides some hints for optimizing, improving convergence

FAF TUTORIAL: SPATIOTEMPORAL PATTERN RECOGNITION (SPR)

Hecht-Nielsen

- OVERVIEW
- ARCHITECTURE
- NEURALWORKS IMPLEMENTATION

OVERVIEW

- Network inputs/outputs are explicit functions of time

- Network transforms the input pattern x(t) into a time-varying class output y(t).

- Network output at t depends on current and previous inputs

- Two basic types: pattern classification / control

- Example of pattern classifier in the speech domain: Given an input stream with objects (words) in it, the output is the class to which the most recently recognized word belongs

- Example of control: the components of x are the system state variables (plant sensor outputs) and the components of y are the plant control signals. The goal is to maximize performance by minimizing some cost functions.

- Goal of SPR: to develop networks that are insensitive to certain transformation of the input patterns

- Want to know ways to measure the distance between 2 patterns

- SPR pattern is a trajectory or path in n-dimensional space, parameterized by time

OVERVIEW

• Typical goal: provide a classification for a relatively brief s-t pattern: the classification occurs after the entire pattern has been entered into the system

• CUEING:
A CUED CLASSIFIER is told when the input pattern begins/ends. In speech this is known as the "isolated word recognition problem"; pauses between words can be detected; therefore the words can be isolated.

AN UNCUED CLASSIFIER deals with a continuous stream of s-t pattern input. IT must figure out when/where the pattern begins / ends.

Two problems in uncued patterns: obscuration and interference.
Obscuration: patterns of interest are obscured by other elements
Interference: for example, mixing sounds from different sources

ASSUME: No obscuration or interference; otherwise problem is intractable.

OVERVIEW

• SPATIOTEMPORAL WARPING: transformation of s-t pattern. S-t pattern classifiers must be insensitive to warping transformations.
1. Time Warp: speeds up or slows down the movement of pattern x along its trajectory (translates it forward or backward in time)
Pattern still traverses the same trajectory, but at a different speed
Ratio of speeds before/after warping is d0/dt where 0 is a monotonically increasing smooth scalar function of time x(0(t))
2. Entire path changes (example in speech, the pitch changes)

• In principle, an s-t pattern of finite duration , not subjected to s-t warping transformations, can be treated as a spatial pattern.
• An s-t warped version of a pattern can be viewed as a different spatial pattern of the same class as the original
• if a time window of a fixed number N of spatial samples is used, the total pattern time durations can sometimes be ignored. "Time vignettes" each classified individually.

OVERVIEW

S-T Pattern Distance Measurement uses matched filter:

$$H_V(u,t) \equiv \inf_{T \in C} \int_{-\infty}^{\infty} \mu(\tau - t) \left| u(\tau) - Tv(\tau) \right| d\tau$$

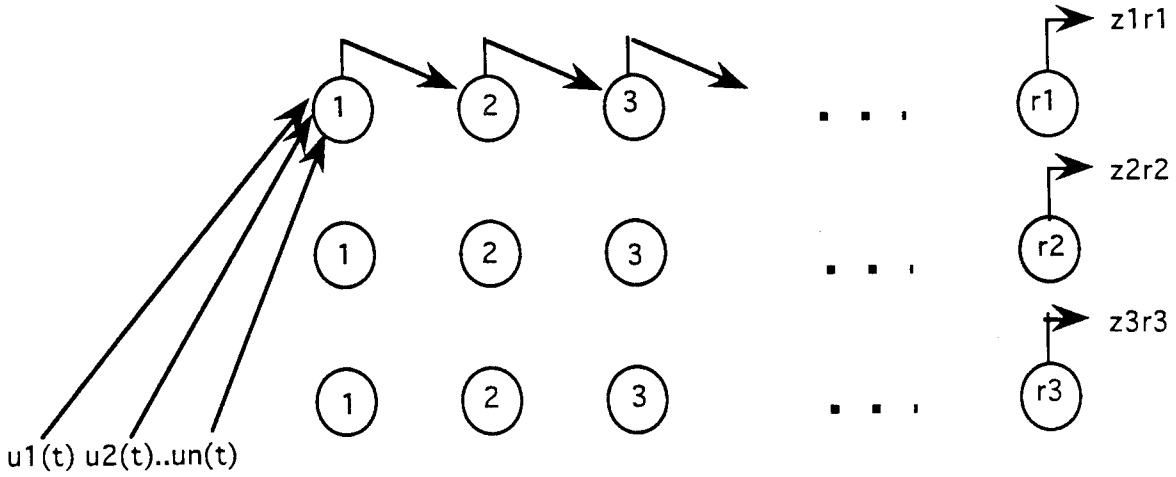
where μ is a time windowing function, focuses the distance measurement on the time interval [t-a,t]. H is the distance between pattern u and the best matching warped portion of v, over the time interval [t-a,t].

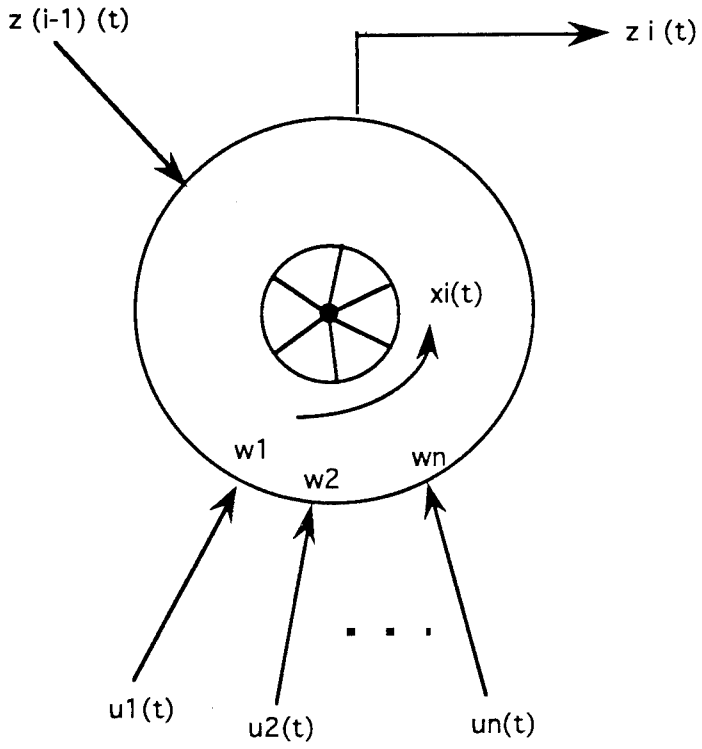
- Nearest Matched Filter Classifier:
  - Given a training set of patterns P = (v1, b1),(v2,b2),...,(vn,bn) where bk is an element of {1,2,...,M}, the set of pattern classifications.
  - Use the training set patterns as the reference patterns for N matched filters. The input pattern u is fed to all the matched filters in parallel. All use the same warping function. The outputs of the classifier at time t are (1) the class # associated with the reference pattern having the smallest matched filter output, and (2) the actual filter output value.

- Problem: ENORMOUS TRAINING SET (many pattern examples!)

- Advantages: near Bayesian performance; individual matched filters are insensitive to noise

ARCHITECTURE





SPR processing element (node). Input comes from previous processing element in row, as well as from pattern u(t). These inputs cause a "braked flywheel" to spin up. The output zi of the unit is 1 if the flywheel is spinning faster than threshold T; otherwise the output is 0.

ARCHITECTURE

- Each row implements a matched filter function for the training set reference pattern.

- t is an integer variable; time increment chosen to be small.

- Basic idea: output of the final processing element of one row should be a binary indicator of whether or not the s-t pattern u has just completed aproximately traversing the path in space defined by the s-t example pattern v (in the proper direction and at a speed within selected time warp limits of the speed of v at each point in the trajectory.

EQUATIONS (Hecht-Nielsen)

MATCHED FILTER: suitable for single dimensional signals. u=input scalar, tuned to scalar signal v

$$H_v(u,t) = \int_{-\infty}^{\infty} \mu(\tau - t)\, v(\tau)\, d\tau$$

GENERALIZED (to n-dimensional signals) MULTIDIMENSIONAL MATCHED FILTER: s-t pattern u, tuned to s-t pattern v, over warp class C:

$$H_v(u,t) \equiv \inf_{T \in C} \int_{-\infty}^{\infty} \mu(\tau - t)\, \left| u(\tau) - Tv(\tau) \right|\, d\tau$$

where μ is a time windowing function, focuses the distance measurement on the time interval [t-a,t]. H is the distance between pattern u and the best matching warped portion of v, over the time interval [t-a,t].

EQUATIONS (Hecht-Nielsen)

SPATIOTEMPORAL PATTERN RECOGNIZER NETWORK: approximately implements a type of nearest matched filter classifier

C = $\theta(t)$ for which $0.5 \leq d\theta/dt \leq 2.0$

Time window = time length of pattern (with total time integrals of 1.0)

Transfer Function: $z_{ji} = U(x_{ji}(t) - \sigma_{ji})$

where

$$x_{ji}(t) = \alpha_{ji}(-c_{ji}x_{ji}(t-1) + d_{ji}\, U([\Psi_{ji} - |v_{ji} - u(t)|]\, z_{j(i-1)}(t-1)))$$

$0 \leq x_{ji}(t) \leq 1$

$z_{jo}(t) \equiv 1$

$U(p) = 1$ if $p > 0$, 0 if $p \leq 0$

$\alpha_{ji}(q) = q$ if $q \geq 0$, Φ if $q < 0$

v = constant vector, c and $\Phi < 1$

c, d, Φ determine flywheel dynamics, matched to typical range of change rates of u & v patterns

σ = threshold

1/c controls speed of x activation

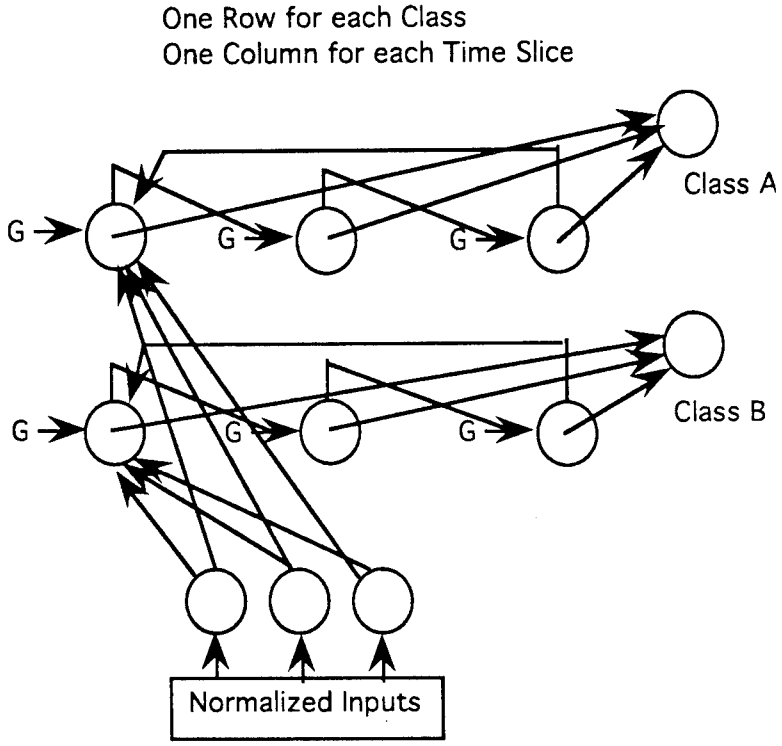
Φ/c controls speed of x decay

Ψ = radius of sphere around v

α = attack function

d = flywheel driving torque

NEURALWORKS IMPLEMENTATION



Avalanche Network.
Inputs must be normalized.
Kohonen learning rule used to adapt weights connected to input layer:
$W' = W + A(X-W)$ where X is input vector, A is learning rate.

Weighted sum I computed in standard fashion. Consists of:

- **Dot product of input vector with associated weight vector (both normalized)**
- **Input from prior processing element in activation chain. This input predisposes the PE to activity.**
- **Global bias term Γ. Used to normalize overall activity in the network. Sets a variable threshold against which PEs compete; assures best match winner.**

NEURALWORKS IMPLEMENTATION

New output computing using I:

$X'_i = X_i + A(-a * X_i + b * [I]^+ - \Gamma + dX^{prev}$

where: x' is the new output

x is the previous output

I is the weighted sum

A(u) is the attack function A(u) = u if u > 0, c*u if u < 0

[u] is a threshold function, = u if u > 0, = 0 if u < 0

a is a decay term for the PE output

b regulates the importance of a new input

c controls the delay of the attack function

d is the amount of pre-condition from prior PE

Γ calculation:

$S = \Sigma x$

$\Gamma = \text{Max}\ (\Gamma + d * (e - T) + f * s\ ,\ 0)$

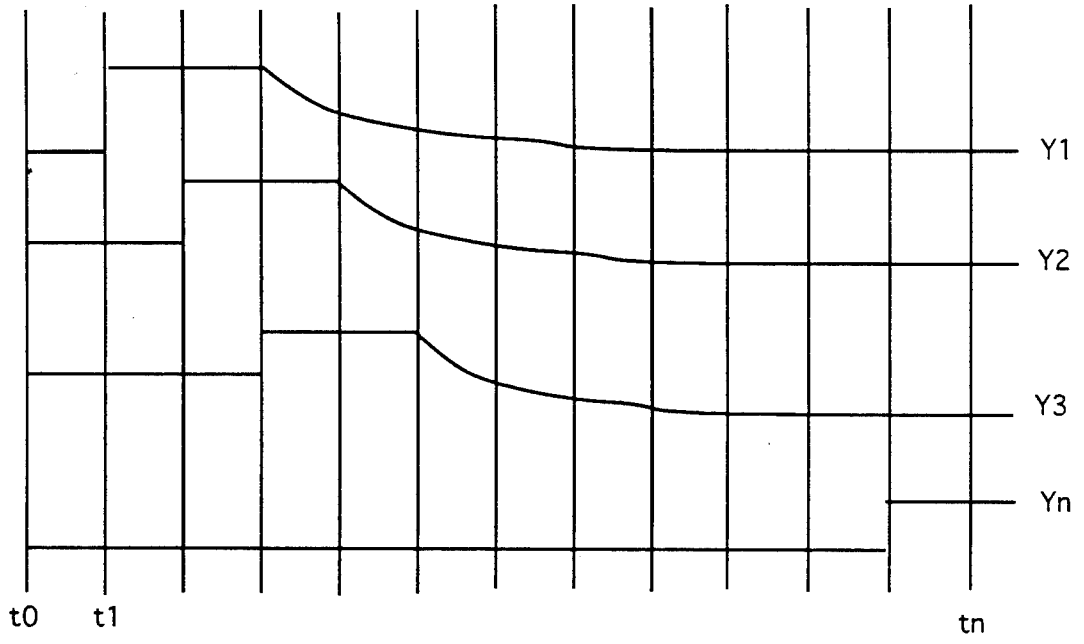
where: S is total network activity

s is change in total network activity

T is a threshold or target power level.

NEURALWORKS IMPLEMENTATION

Avalanche of activity through the chain:
Y1 through Yn represent the activity of succeeding PEs in detection chain.



2. APPENDIX J. BIT/FAULT REPORT CAUSE TUTORIALS

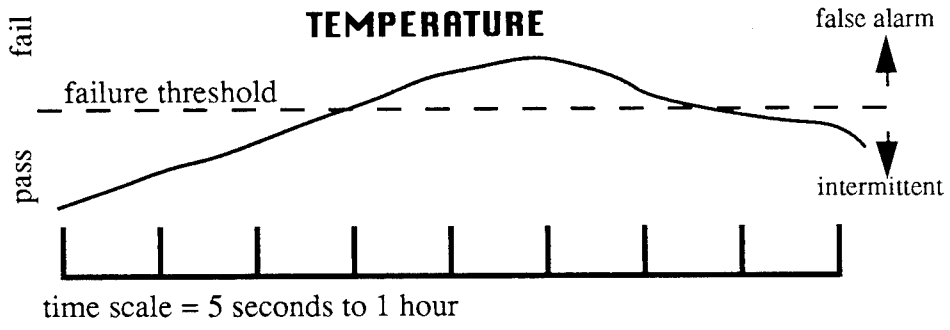
This appendix contains the BIT and fault report cause tutorials which were held throughout the NNFAF contract. A tutorial was held for each of the BIT techniques and fault report causes which were selected for the NNFAF demonstration approaches. The BIT techniques were error correcting (Viterbi), activity detection, and parity. The fault report causes were temperature, vibration, and G-load.

BIT TECHNIQUE and FAULT REPORT CAUSE TUTORIAL

Error Correcting BIT with Temperature Fault Report Cause

****

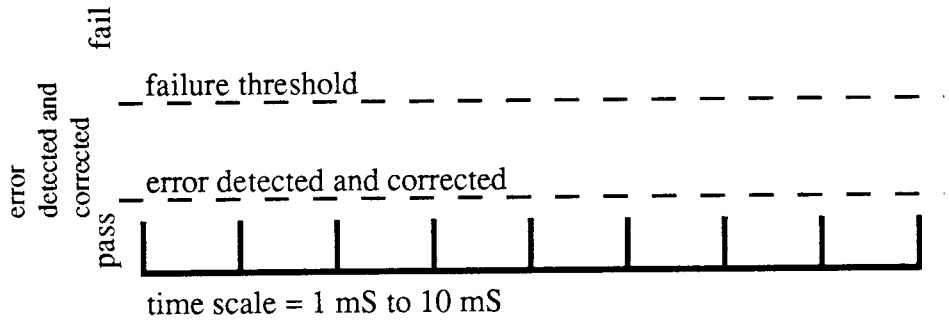
Fault Report Cause



•Threshold line represents typical hardware response to the temperature curve. Hardware will pass BIT tests when the temperature curve is below the threshold and will fail BIT tests when the temperature curve is above the threshold.

• An individual system's hardware will exhibit the same temperature curve/threshold relationship, but the threshold (of BIT failure) will vary. The threshold depicted above shows the boundary between an intermittent failure zone and a false alarm zone. The BIT reports of any system that responds to temperature with a threshold above the threshold in the figure will be false alarm signatures. BIT reports that respond to temperature with a threshold below the threshold in the figure will be intermittent failure signatures.

Error Correcting BIT Signatures



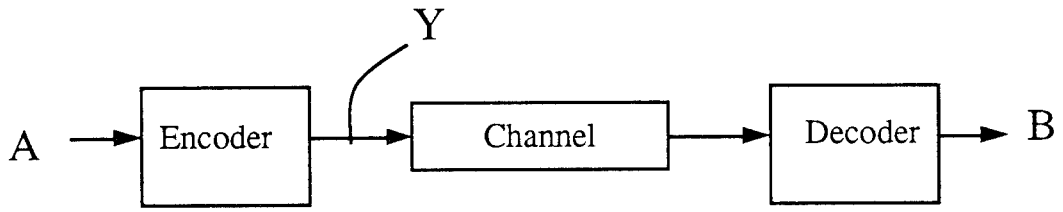
- Error correcting BIT techniques provide reports with three states:

1. No error
2. Error detected and corrected
3. Error detected but not correctable

- These three states provide enhanced information compared to pass/fail reports from a typical BIT technique.

- The error detected and corrected threshold is located relative to a fault report cause curve the same as a pass/fail threshold. The error detected but not correctable threshold provides additional information in the BIT report. It is a more severe failure than the error detected and corrected failure.

Error Correcting BIT Techniques



Input Data
$A = (A_1, A_2, ..., A_m)$
$A_i = (a_{i1}, a_{i2}, ..., a_{in})$

Output Data
$B = (B_1, B_2, ..., B_m)$
$B_i = (b_{i1}, b_{i2}, ..., b_{in})$

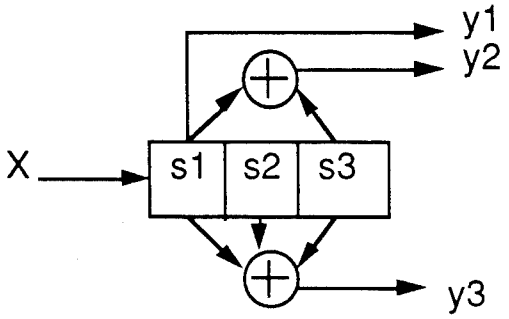
A = transmit data block
Ai = one data word
m = # of input words
n = bits in data word

Two types of techniques:
1. Block Code: Data word is independent of other data words.
2. Convolutional Code: Data word is dependent on other data.

Convolutional Encoding

- Uses shift register to accept inputs and generate outputs.

Ex.



Outputs
Yi = (y1, y2, y3)

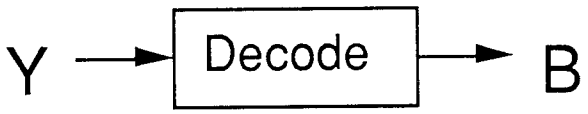
Internal State
S = (s1, s2, s3)

- Y = F(s1, s2, s3) where input x = s1
- Therefore, Yt = F(xt, xt-1, xt-2)

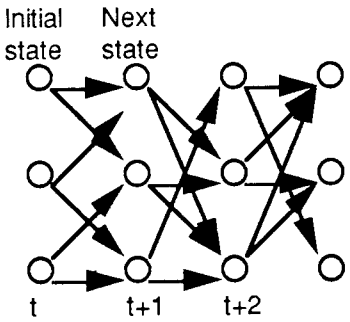
Y is a function of the present and past 2 input states.

Convolutional Code Decoding/Viterbi

•Dynamic Programming: finds minimum distance between received code word and possible code words.



•Trellis Diagram Example:



Each line represents a valid path to the next state

Valid paths determined by algorithm

Circles represent states

•When decoding - distance to each state is determined. The shortest distance path is chosen as the correct data. If the shortest distance path is zero, then no error occurred.

•The number of previously received data states (j) used to determine valid data, is defined by the algorithm.

•Bt = F(yt, yt-1, ..., yt-j) j = depth of algorithm

Block Codes

HAMMING CODE:

- Additional bits are added to data bits to generate a valid code word.

- Modulo 2 matrix multiplication can be applied to the code words to detect errors.

- Hamming codes are defined by two parameters (n, k) where n is the total number of code word bits and k is the number of data bits.

- Example: (7, 3) Hamming code. Data = (D1, D2, & D3), Parity = (P1, P2, P3, & P4)

- The capabilities of error detection and correction codes are related to the minimum difference between valid code words, referred to as the Hamming distance. This number is equal to the number of 1's resulting from XORing two code words.

(7, 3) Hamming Code Example

$P_1 = D_1 \text{ XOR } D_3$
$P_2 = D_1 \text{ XOR } D_2$
$P_3 = D_2 \text{ XOR } D_3$
$P_4 = D_1 \text{ XOR } D_2 \text{ XOR } D_3$

Total set of valid code words:

Data bits (D_1, D_2, & D_3)			Parity bits (P_1, P_2, P_3, & P_4)			
0	0	0	0	0	0	0
0	0	1	1	0	1	1
0	1	0	0	1	1	1
0	1	1	1	1	0	0
1	0	0	1	1	0	1
1	0	1	0	1	1	0
1	1	0	1	0	1	0
1	1	1	0	0	0	1

- Hamming distance for this example is 4. If 1, 2, or 3 bits are corrupted then the corrupted word will not match any code word.

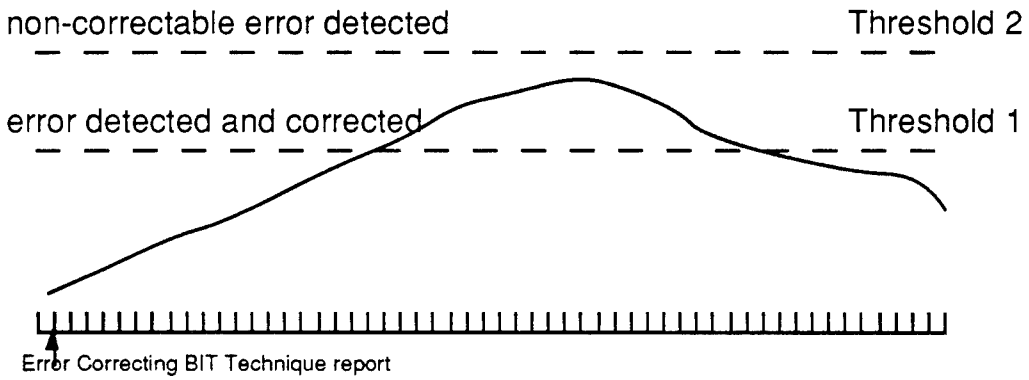
If 1 bit is corrupted, then only one valid word will have a Hamming distance from the corrupted word of 1 and all others will be larger. That valid word is the correct word if only 1 or 2 bit errors are expected.

For example, if the code word 010 0111 is corrupted to form 011 0111, then the Hamming distance from valid words is as follows:

Total set of valid code words:	Hamming distance from 011 0111
000 0000	5
001 1011	3
010 0111	1
011 1100	3
100 1101	5
101 0110	3
110 1010	5
111 0001	3

010 0111 is the valid code word because only 1 or 2 bits were assumed to be erroneous. If two bits are corrupted, then the (7, 3) code can detect the error but several valid words may have a distance of 2 and the error cannot be fixed.

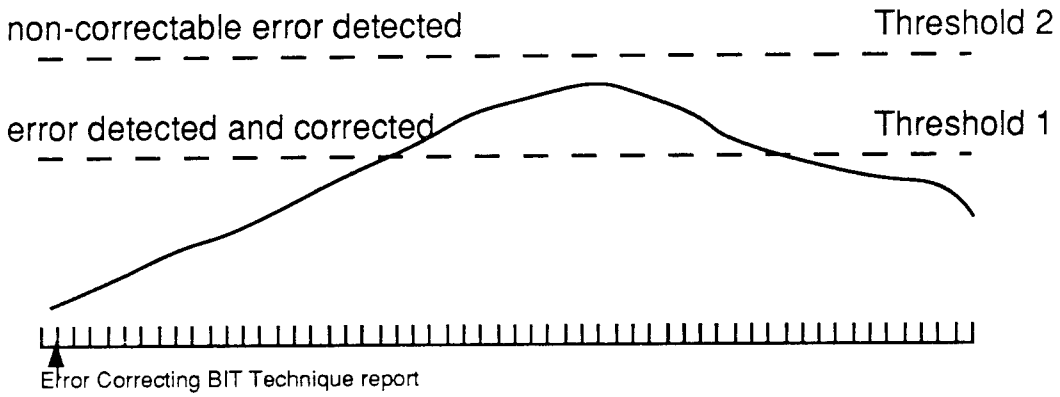
Signature of Hamming Code BIT Technique with Temperature Fault Report Cause



Failure detection states: 0: No error
1: Error detected and corrected
2: Error detected but not correctable

Threshold 1 and 2 will vary relative to each other for different systems. If they are lower than the thresholds shown then they represent a system with an intermittent failure. If they are above the thresholds shown then the system may only have a false alarm.

Hamming Code BIT Report with Temperature



BIT Reports:

Functional system with thresholds above depicted thresholds:
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0

Functional system with false alarm with threshold as shown above:
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 1 1 1 0 1 0 1 1 1 1 0 0 0 0 0 0 0

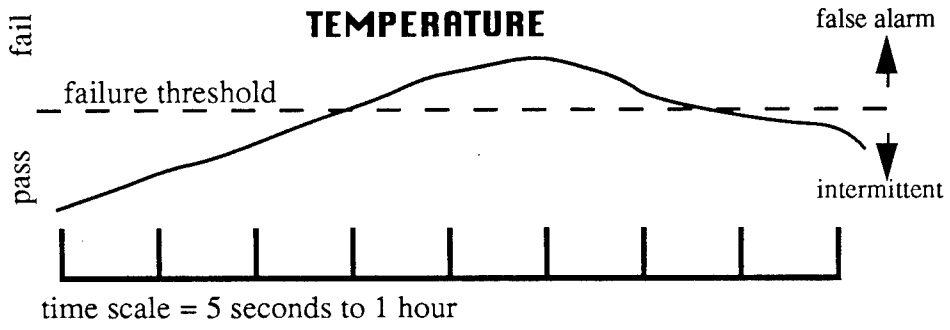
System with intermittent failure (thresholds lower than shown above):
0 0 0 0 0 0 0 0 0 1 1 1 1 0 1 1 2 1 2 2 1 0 2 2 2 1 1 0 1 1 1 0 1 0 1 0 0 0

BIT TECHNIQUE and FAULT REPORT CAUSE TUTORIAL

Activity Detection BIT with Temperature Fault Report Cause

****

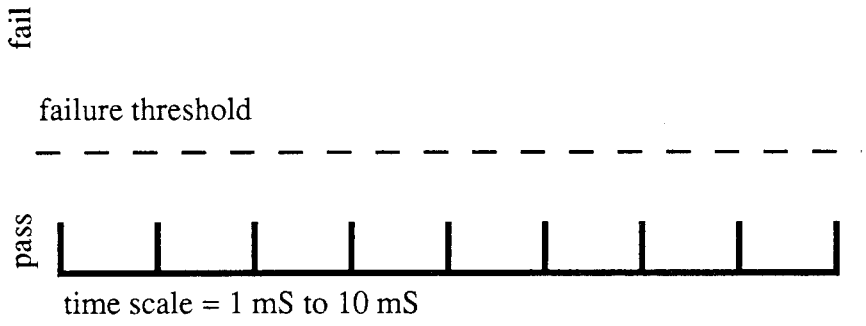
Fault Report Cause



•Threshold line represents typical hardware response to the temperature curve. Hardware will pass BIT tests when the temperature curve is below the threshold and will fail BIT tests when the temperature curve is above the threshold.

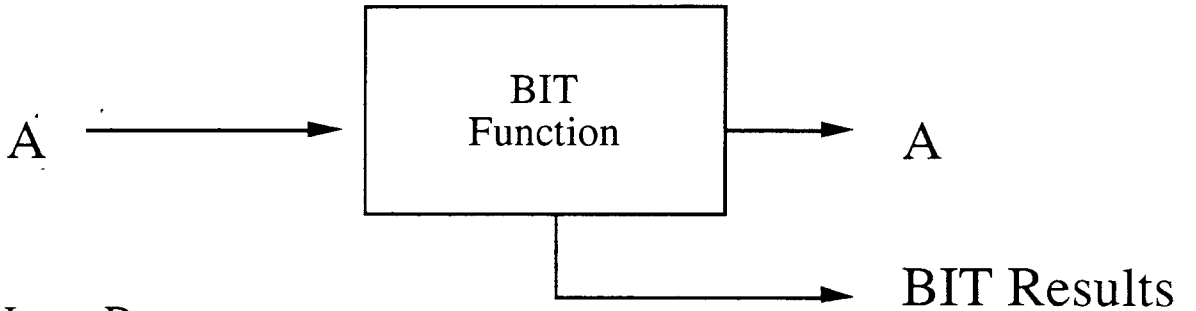
•An individual system's hardware will exhibit the same temperature curve/threshold relationship, but the threshold (of BIT failure) will vary. The threshold depicted above shows the boundary between an intermittent failure zone and a false alarm zone. The BIT reports of any system that responds to temperature with a threshold above the threshold in the figure will be false alarm signatures. BIT reports that respond to temperature with a threshold below the threshold in the figure will be intermittent failure signatures.

Pass/Fail BIT Signatures



- **Pass/Fail BIT techniques provide reports with two states:**
  1. **No error**
  2. **Error detected**

Activity Detector BIT Techniques



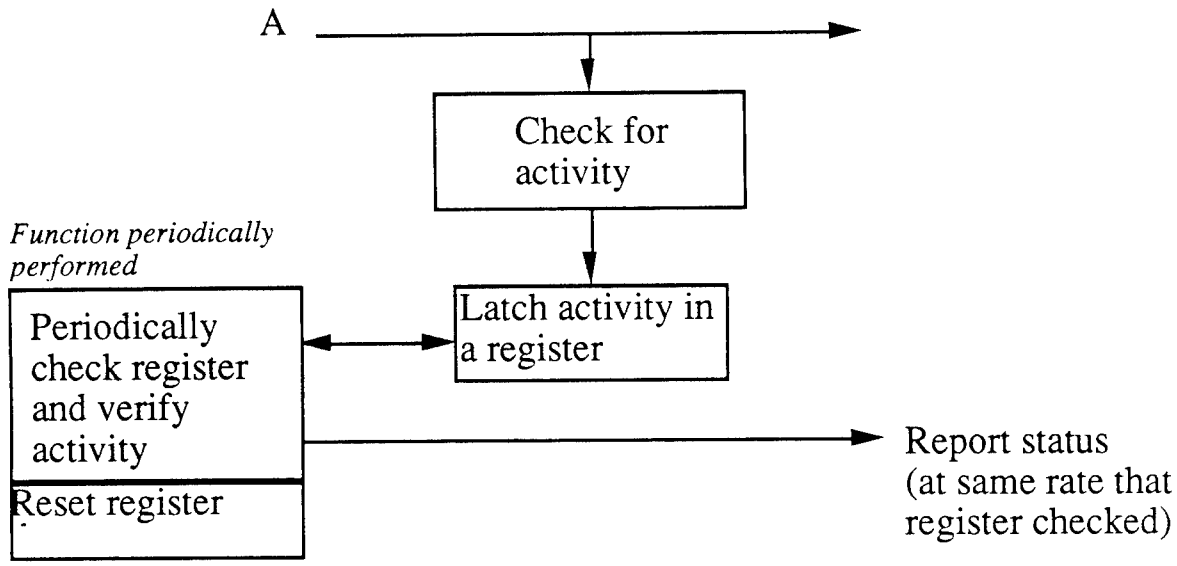
Input Data
$A = (A_1, A_2, ..., A_m)$
$A_i = (a_{i1}, a_{i2}, ..., a_{in})$

A = transmit data block
Ai = one data word
m = # of input words
n = bits in data word

Output Data is
unaltered input data

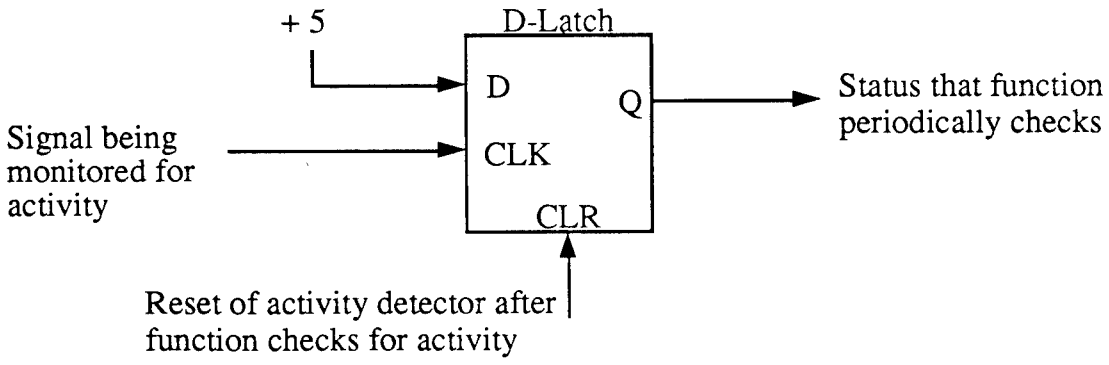
Activity Detector

Signals are constantly monitored for state changes. Once a state change occurs activity is triggered and recorded in a register for the respective signal. Periodically, the activity detector status register is checked and reset. If any status register signals do not confirm that activity occurred, then a failure is reported.



Activity Detector Example

An implementation of an activity detector latch is shown below:



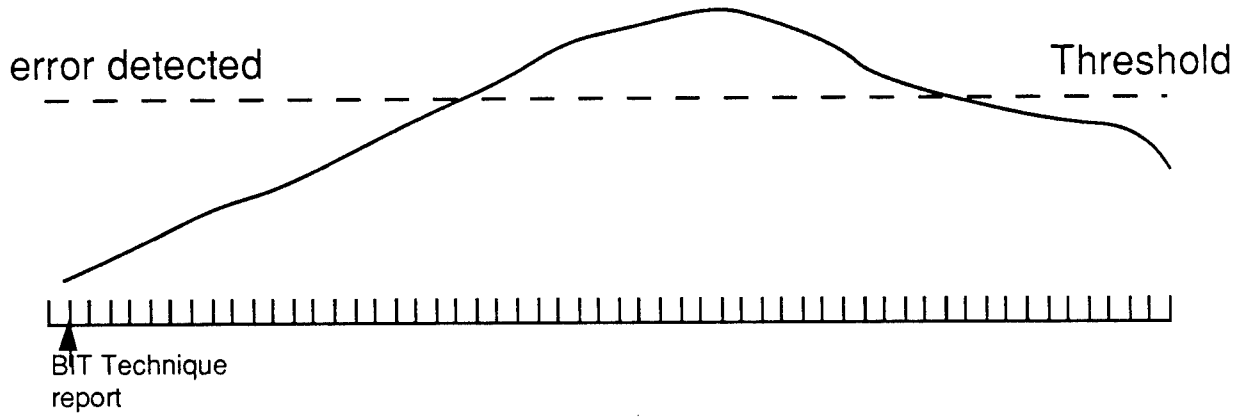
This example of an activity detector is triggered by a low to high state signal transition.

Activity Detector Example

Example of activity detection using the activity detector from the previous page

Input data (D1, D2, D3, D4)	Activity Status Register (1 = fail) (S1, S2, S3, S4)	Periodic Check of Activity Register (0 = P)	
0 0 0 0	0 0 0 0	X	
0 0 1 0	0 0 1 0	X	
0 1 1 1	0 1 1 1	X	
0 1 0 0	0 1 1 1	X	
1 0 0 1	1 1 1 1	X	
0 0 1 0	1 1 1 1	0	Pass Report
0 0 0 0	0 0 0 0	X	
0 0 0 1	0 0 0 1	X	
1 0 0 0	1 0 0 1	X	
0 1 0 0	1 1 0 1	X	
0 1 0 1	1 1 0 1	X	
1 1 0 0	1 1 0 1	1	Fail Report
0 0 1 0	0 0 1 0	X	

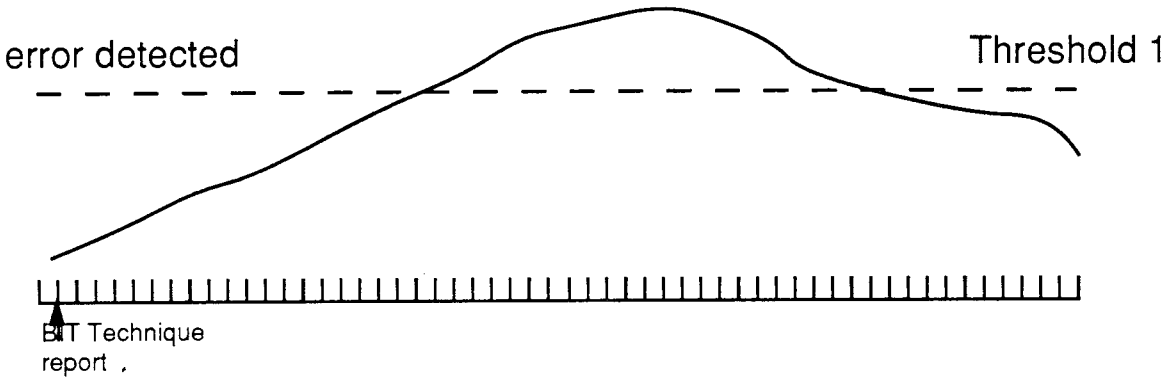
Signature of Activity Detection BIT Technique with Temperature Fault Report Cause



Failure detection states: 0: No error
1: Error detected

The threshold will vary for different systems. If it is lower than the threshold shown then it represents a system with an intermittent failure. If it is above the threshold shown then the system may only have a false alarm.

Activity Detection BIT Report with Temperature



Possible BIT Reports in response to the curve above:

Functional system with thresholds above depicted thresholds:
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0

Functional system with false alarm with threshold as shown above:
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 1 1 1 0 1 0 1 1 1 1 1 0 0 0 0 0 0 0 0

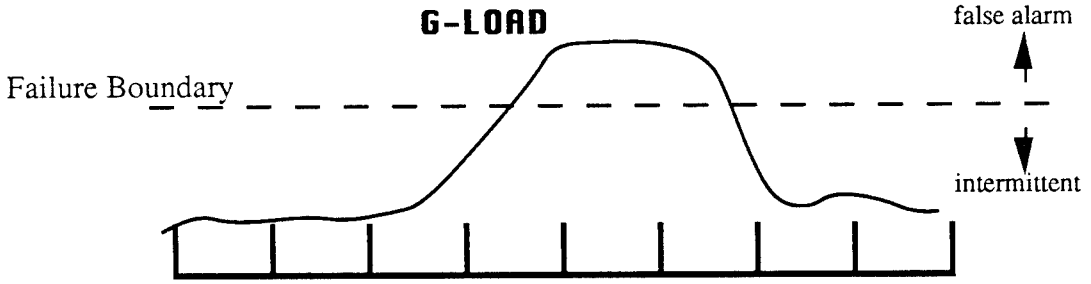
System with intermittent failure (thresholds lower than shown above):
0 0 0 0 0 0 0 0 0 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 0 1 0 1 0 0 0

BIT TECHNIQUE and FAULT REPORT CAUSE TUTORIAL

Parity BIT with G-Load Fault Report Cause

****

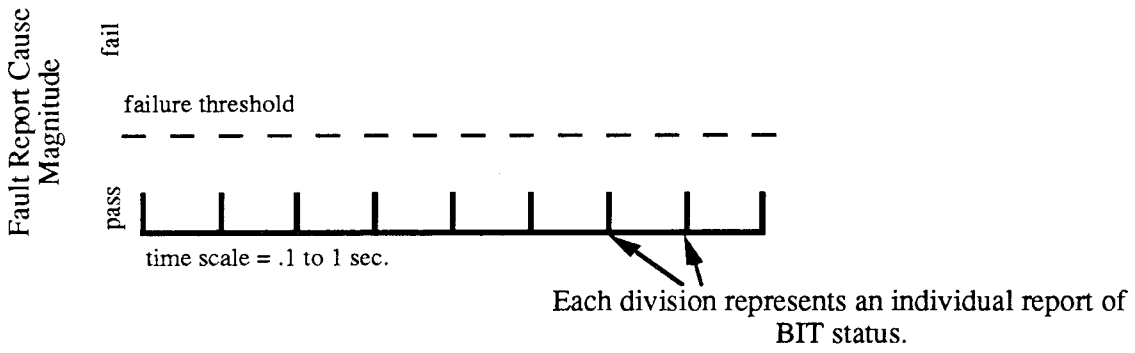
Fault Report Cause



•Boundary line represents typical hardware response to the environment curve. Hardware is more likely to pass BIT tests when the environment curve is below the boundary and will fail BIT tests when the environment curve increases above the boundary.

•An individual system's hardware will exhibit the same environment (G-Load) curve/boundary relationship, but the threshold (of BIT failure) will vary. The figure depicted above shows the boundary between an intermittent failure zone and a false alarm zone. The BIT reports of any system that respond to G-Load with a threshold above the boundary in the figure will be false alarm signatures. BIT reports that respond to G-Load with a threshold below the boundary in the figure will be intermittent failure signatures.

Pass/Fail BIT Reporting

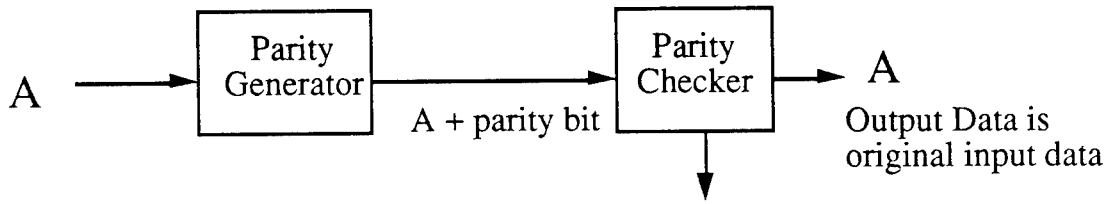


- **Pass/Fail BIT techniques provide reports with two states:**

  1. **No error**
  2. **Error detected**

- **BIT is unlikely to report a failure unless the magnitude of a fault report cause (FRC) is above the failure threshold.**

Parity BIT Techniques

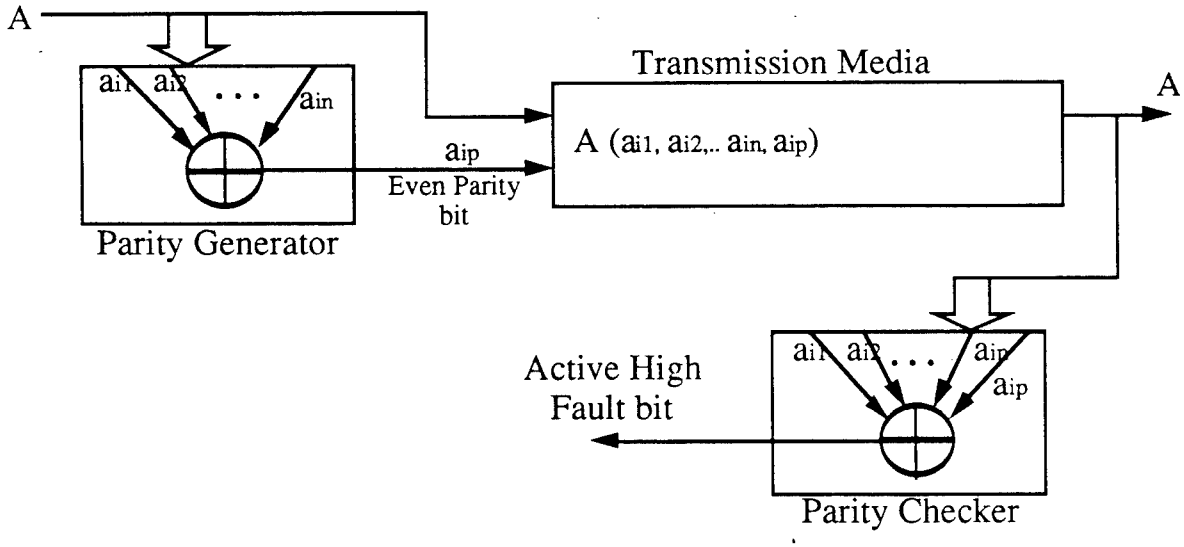


Input Data
$A_i = (a_{i1}, a_{i2}, ..., a_{in})$

Ai = one data word
n = bits in data word

Parity

A parity generator is used to compute a parity (in this example it is a 1 bit even parity). The parity bit is added to the data word prior to being stored or transmitted. When the data word (with parity) is received or read, the correct parity is confirmed.



Parity Generation Example

Input data (Ai1, Ai2, Ai3, ...Ai8)								Parity bit (Aip)
0	0	0	0	0	1	0	0	1
0	0	1	0	0	1	1	0	1
0	1	1	1	0	0	1	1	1
0	1	0	0	0	1	1	1	0
1	0	0	1	1	0	1	1	1
0	0	1	0	1	0	1	1	0
0	0	0	0	0	1	0	0	1
0	0	0	1	0	0	0	1	0
1	0	0	0	1	0	0	1	1
0	1	0	0	1	0	0	1	1
0	1	0	1	1	1	0	1	1
1	1	0	0	1	1	0	1	1
0	0	1	0	0	1	1	0	1

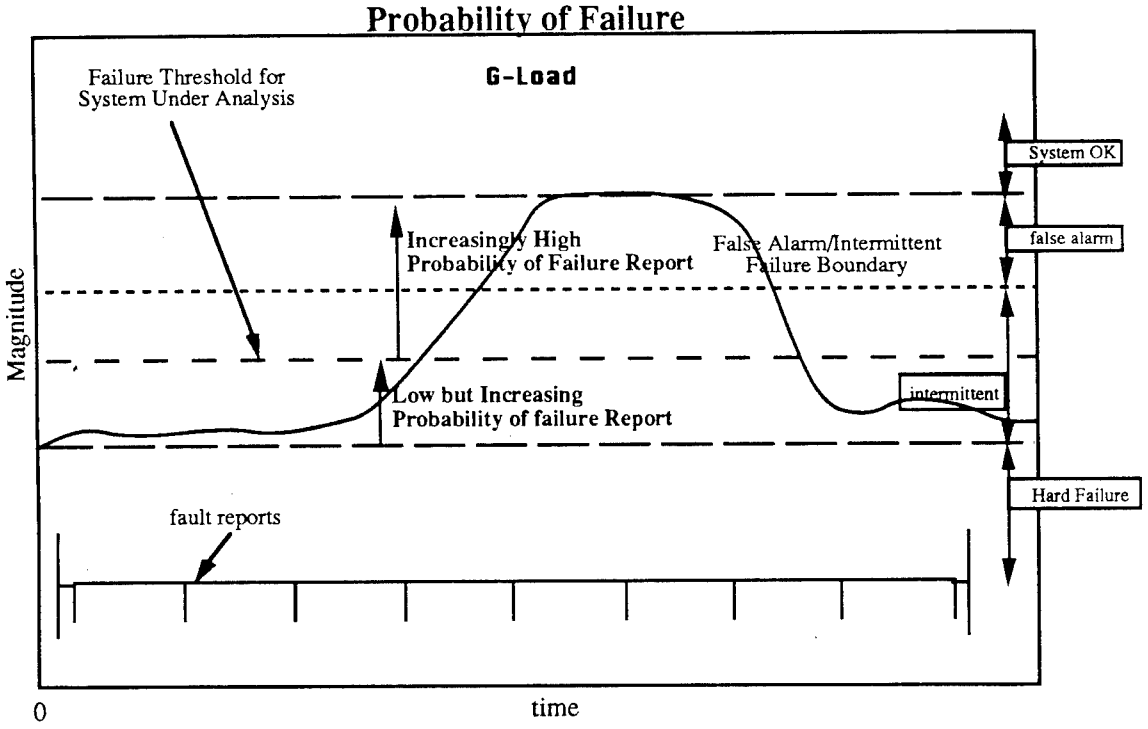
Parity bit generated by using exclusive OR addition of all data bits.

Parity Verification Example

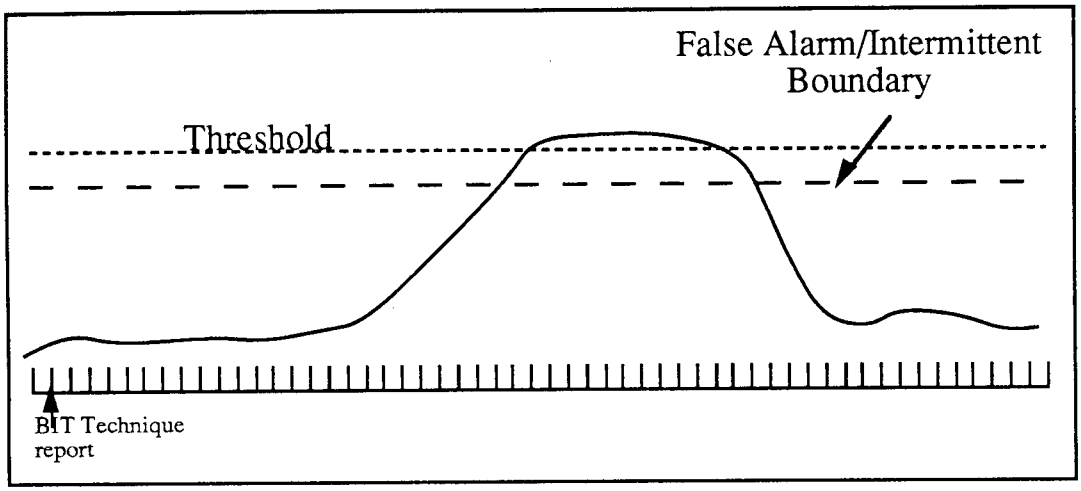
Input data (Ai1, Ai2, Ai3, ...Ai8)								Parity bit (Aip)	Parity test results (fault bit)	
0	0	0	0	0	1	0	0	1	0	
0	0	1	0	0	1	1	0	1	0	*Pass Report*
0	1	**0**	1	0	0	1	1	1	1	
0	1	0	0	0	1	1	1	0	0	Parity test bit generated by using exclusive OR addition of all data bits (including parity bit).
1	0	0	1	1	0	1	1	1	0	
0	0	1	0	1	0	1	1	0	0	
0	0	0	0	0	1	0	0	1	0	
0	0	0	1	0	0	0	1	0	0	
1	0	0	0	1	0	0	1	1	0	
0	1	0	0	1	0	0	1	1	0	
0	1	0	1	1	1	**1**	1	1	1	*Fail Report*
1	1	0	0	1	**0**	0	1	1	1	
0	0	1	0	0	1	1	0	1	0	

Bold type represents corrupted data.

Behavior of Parity BIT Technique with G-Load Fault Report Cause

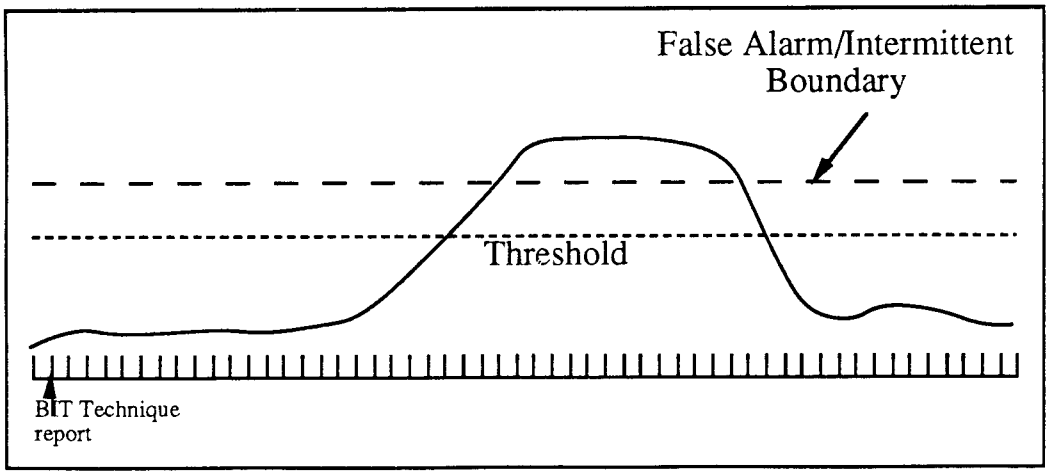


Parity BIT Reports with G-Load Signature



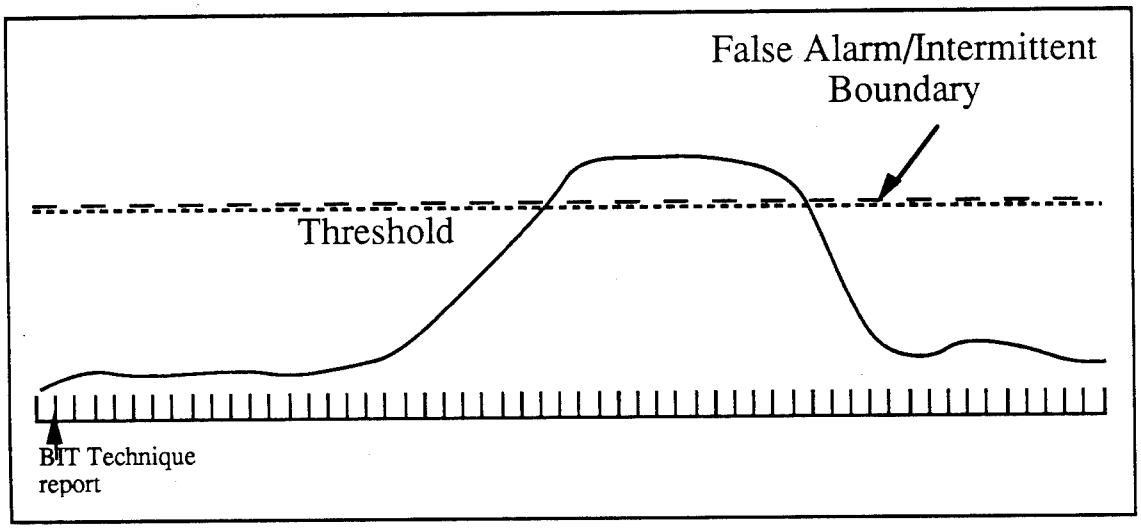
Possible BIT Reports in response to the curve above:

Functional system with threshold above depicted boundary:
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0



System with intermittent failure (threshold lower than boundary):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 1 1 1 0 1 1 1 0 0 0 0 0 0 0 0 0

Parity BIT Report with G-Load Signature



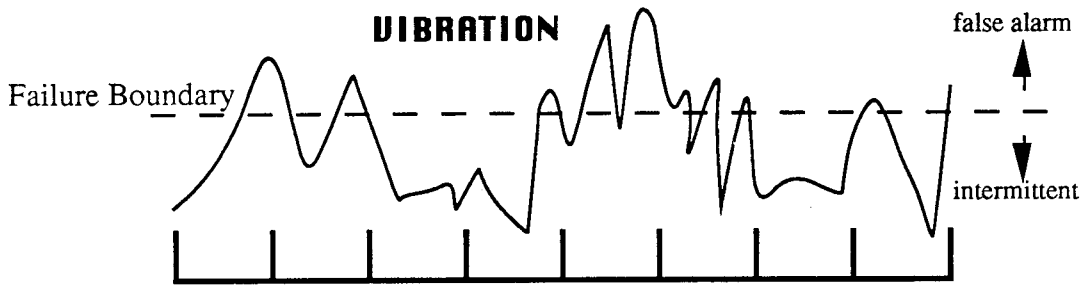
Functional system with threshold at boundary (false alarm):
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 1 1 1 0 0 0 0 0 0 0 0 0

BIT TECHNIQUE and FAULT REPORT CAUSE TUTORIAL

Parity BIT with Vibration Fault Report Cause

****

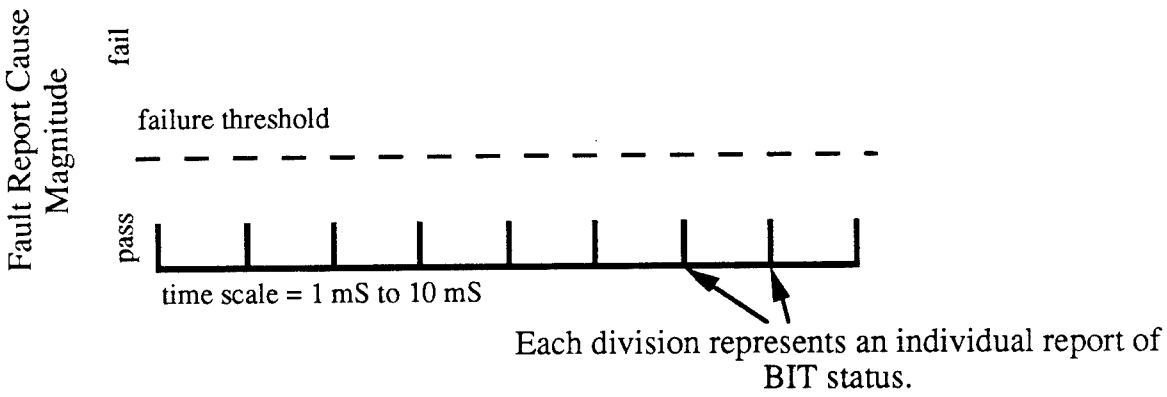
Fault Report Cause



•Boundary line represents typical hardware response to the environment curve. Hardware is more likely to pass BIT tests when the environment curve is below the boundary and will fail BIT tests when the environment curve increases above the boundary.

•An individual system's hardware will exhibit the same environment (vibration) curve/boundary relationship, but the threshold (of BIT failure) will vary. The figure depicted above shows the boundary between an intermittent failure zone and a false alarm zone. The BIT reports of any system that respond to vibration with a threshold above the boundary in the figure will be false alarm signatures. BIT reports that respond to vibration with a threshold below the boundary in the figure will be intermittent failure signatures.

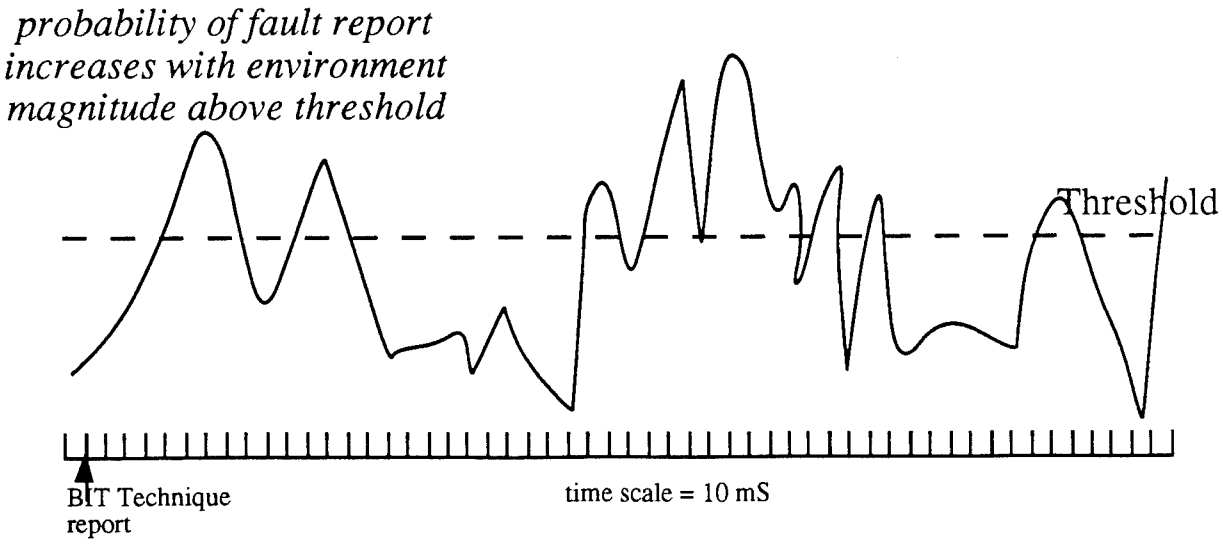
Pass/Fail BIT Reporting



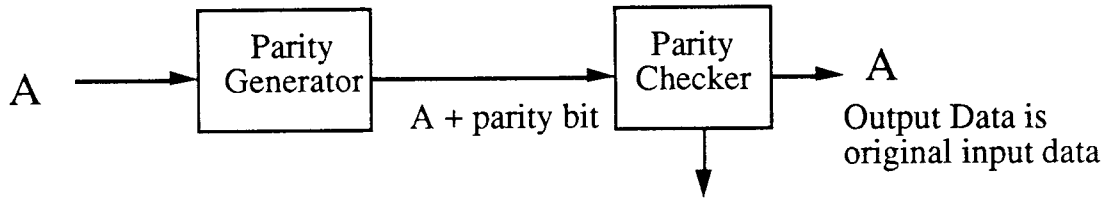
- **Pass/Fail BIT techniques provide reports with two states:**
  1. **No error**
  2. **Error detected**

- **BIT is unlikely to report a failure unless the magnitude of a fault report cause (FRC) is above the failure threshold.**

Fault Report Cause



Parity BIT Techniques

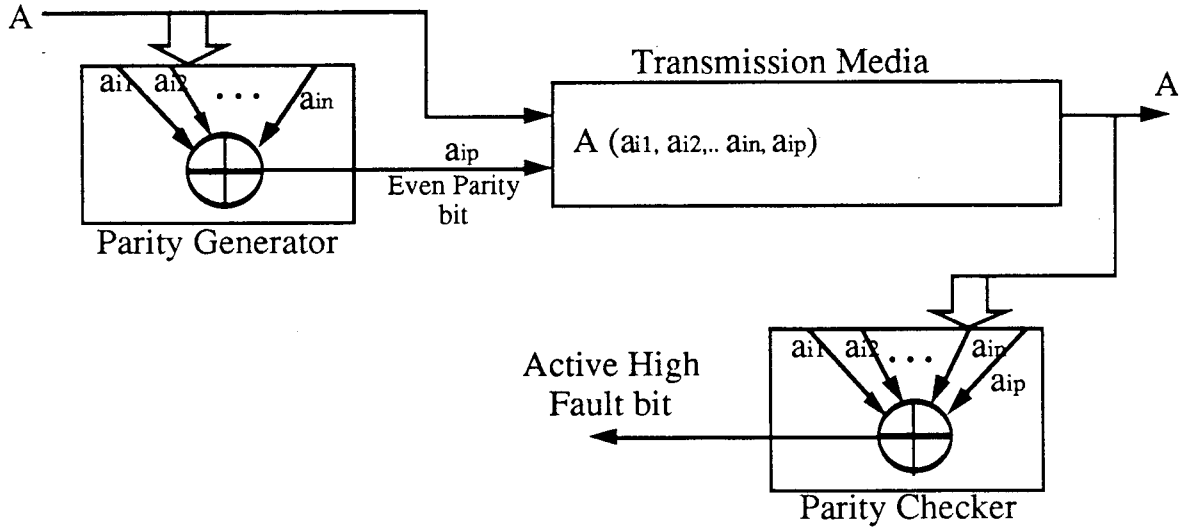


Input Data
$A_i = (a_{i1}, a_{i2}, ..., a_{in})$

Ai = one data word
n = bits in data word

Parity

A parity generator is used to compute a parity (in this example it is a 1 bit even parity). The parity bit is added to the data word prior to being stored or transmitted. When the data word (with parity) is received or read, the correct parity is confirmed.



Parity Generation Example

Input data (Ai1, Ai2, Ai3, ...Ai8)								Parity bit (Aip)
0	0	0	0	0	1	0	0	1
0	0	1	0	0	1	1	0	1
0	1	1	1	0	0	1	1	1
0	1	0	0	0	1	1	1	0
1	0	0	1	1	0	1	1	1
0	0	1	0	1	0	1	1	0
0	0	0	0	0	1	0	0	1
0	0	0	1	0	0	0	1	0
1	0	0	0	1	0	0	1	1
0	1	0	0	1	0	0	1	1
0	1	0	1	1	1	0	1	1
1	1	0	0	1	1	0	1	1
0	0	1	0	0	1	1	0	1

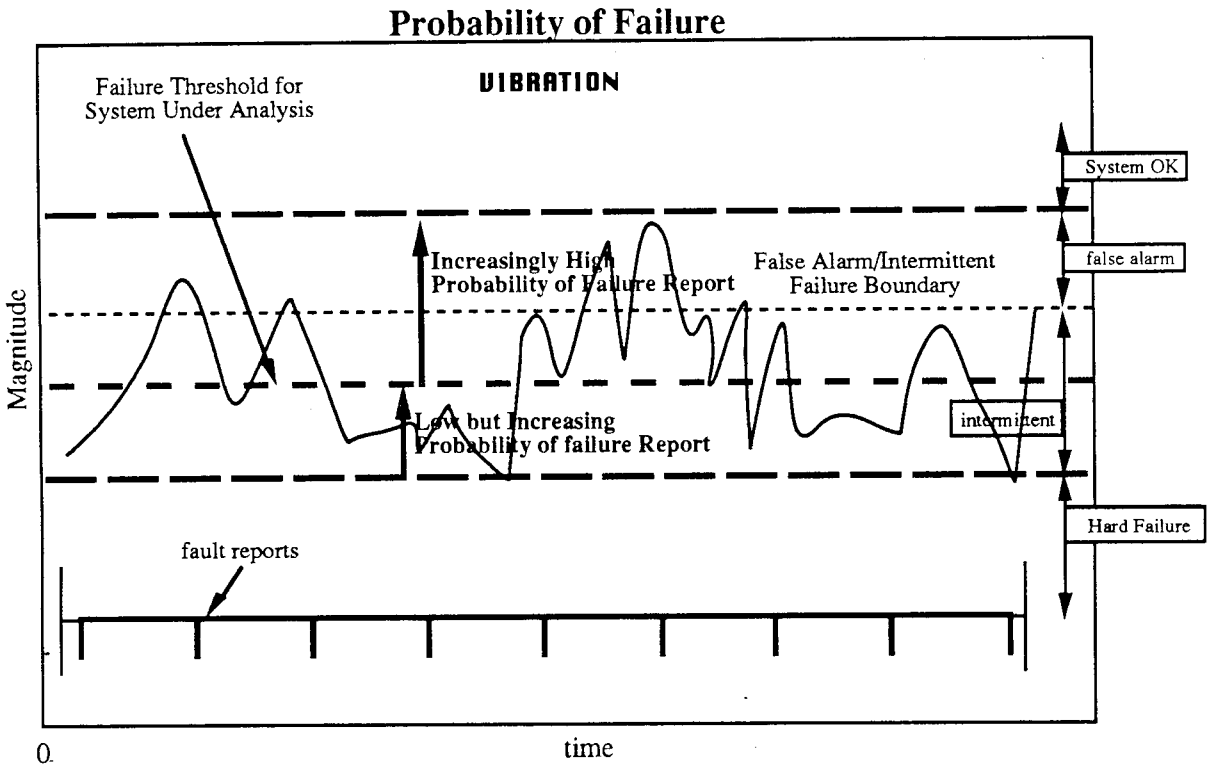
Parity bit generated by using exclusive OR addition of all data bits.

Parity Verification Example

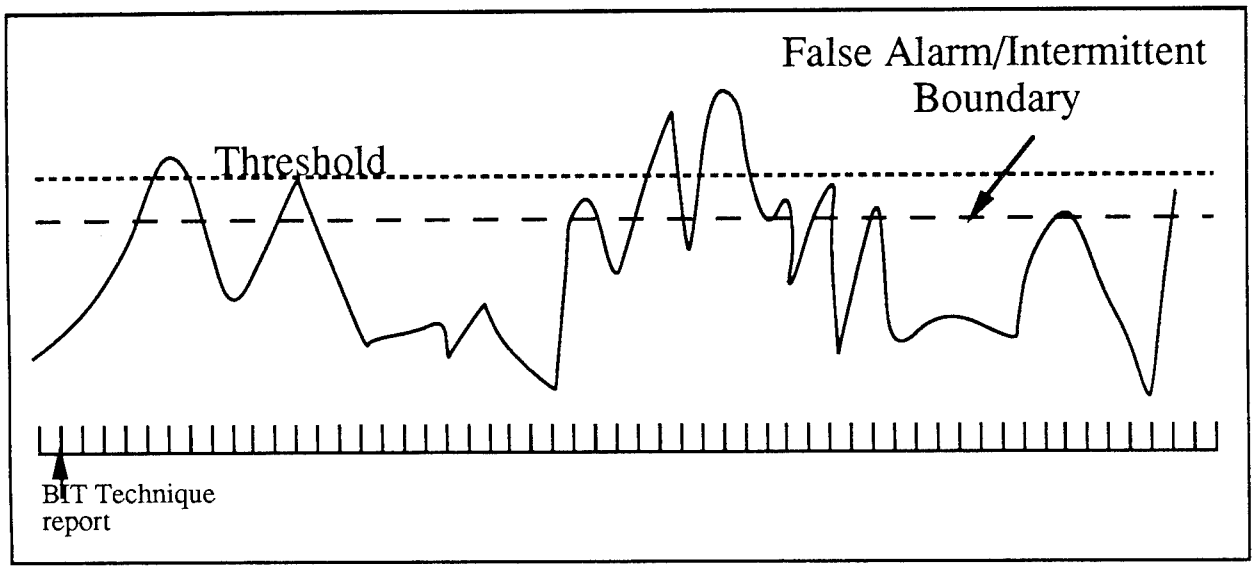
Input data (Ai1, Ai2, Ai3, ...Ai8)								Parity bit (Aip)	Parity test results (fault bit)	
0	0	0	0	0	1	0	0	1	0	
0	0	1	0	0	1	1	0	1	0	*Pass Report*
0	1	**0**	1	0	0	1	1	1	1	
0	1	0	0	0	1	1	1	0	0	Parity test bit generated by using exclusive OR addition of all data bits (including parity bit).
1	0	0	1	1	0	1	1	1	0	
0	0	1	0	1	0	1	1	0	0	
0	0	0	0	0	1	0	0	1	0	
0	0	0	1	0	0	0	1	0	0	
1	0	0	0	1	0	0	1	1	0	
0	1	0	0	1	0	0	1	1	0	
0	1	0	1	1	1	**1**	1	1	1	*Fail Report*
1	1	0	0	1	**0**	0	1	1	1	
0	0	1	0	0	1	1	0	1	0	

Bold type represents corrupted data.

Behavior of Parity BIT Technique with Vibration Fault Report Cause



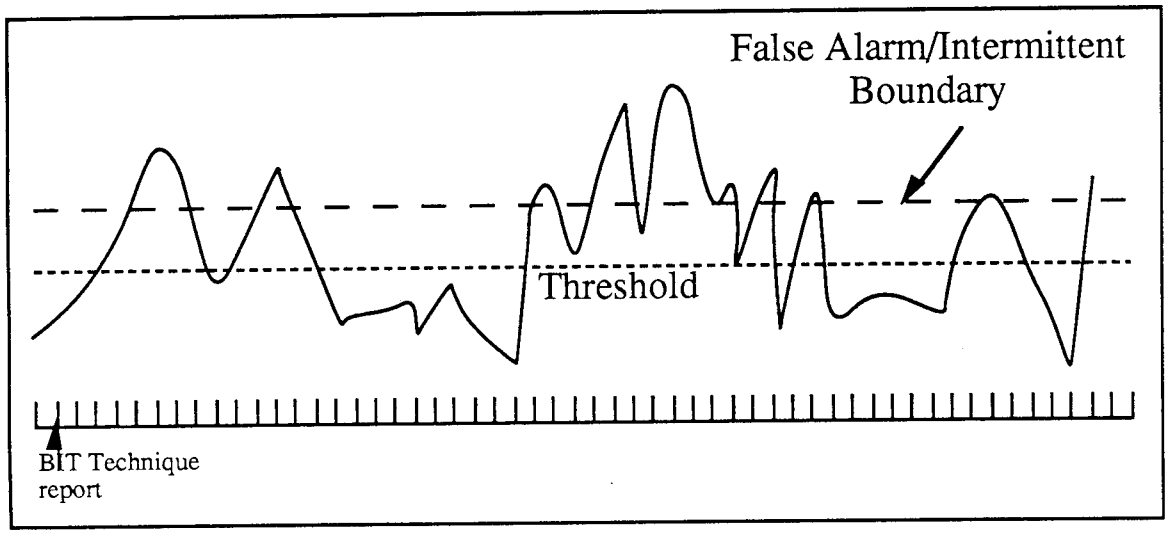
Parity BIT Report with Vibration Signature



Possible BIT Reports in response to the curve above:

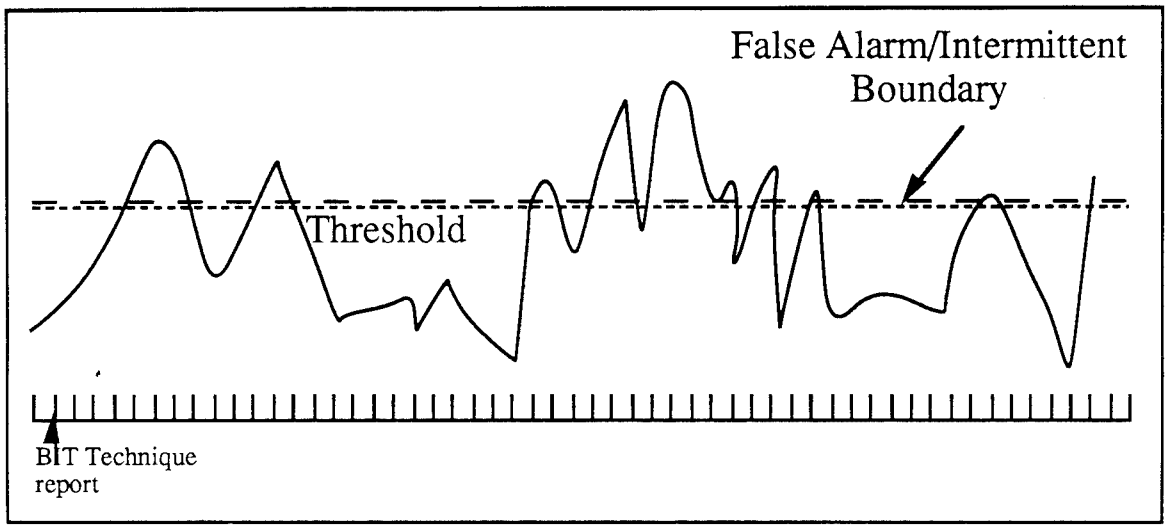
Functional system with threshold above depicted boundary:
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0

Parity BIT Report with Vibration Signature



System with intermittent failure (threshold lower than boundary):
0 0 1 1 0 1 0 1 1 1 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 1 0 0 0 0 0 0 1 1 0 0 1

Parity BIT Report with Vibration Signature



Functional system with threshold at boundary (false alarm):
0 0 1 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 1 1 0 1 1 0 1 0 1 0 0 0 0 1 0 0 0 1

*U.S. GOVERNMENT PRINTING OFFICE: 1995-610-126-50115

MISSION

OF

ROME LABORATORY

Mission. The mission of Rome Laboratory is to advance the science and technologies of command, control, communications and intelligence and to transition them into systems to meet customer needs. To achieve this, Rome Lab:

a. Conducts vigorous research, development and test programs in all applicable technologies;

b. Transitions technology to current and future systems to improve operational capability, readiness, and supportability;

c. Provides a full range of technical support to Air Force Materiel Command product centers and other Air Force organizations;

d. Promotes transfer of technology to the private sector;

e. Maintains leading edge technological expertise in the areas of surveillance, communications, command and control, intelligence, reliability science, electro-magnetic technology, photonics, signal processing, and computational science.

The thrust areas of technical competence include: Surveillance, Communications, Command and Control, Intelligence, Signal Processing, Computer Science and Technology, Electromagnetic Technology, Photonics and Reliability Sciences.